SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD-A172 687

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 86-0904	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Signal Processing in Ultrasonic NDE		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report 8/84 - 11/85
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) P. K. Bhagat, Ph.D.; B. J. Leon, Ph.D.		8. CONTRACT OR GRANT NUMBER(s) AFOSR-84-0223
9. PERFORMING ORGANIZATION NAME AND ADDRESS Biomedical Engineering Center, Wenner-Gren Research Laboratory, University of Kentucky, Lexington, KY 40506-0070		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2306/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research (AFOSR)		12. REPORT DATE July 1986
		13. NUMBER OF PAGES 91
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) AFOSR Bldg 410 BAFB DC 20332		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

DTIC ELECTE OCT 0 8 1986 S D

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

D

18. SUPPLEMENTARY NOTES

Dr. J. Hager (AFOSR)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Ultrasound, A/D Conversion, Wiener Filter, Spline Functions, Homomorphic Filters, Coherent Noise

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The desirable performance goal of an ultrasonic nondestructive evaluation, NDE, methodology is to reliably and rapidly obtain information regarding flaws in the material being tested. Decisions concerning acceptance/rejection of material for further usage can then be made on the basis of the nature and severity of flaws within it. Flaw size estimates are currently made on the basis of an idealized physical model, describing the flaw, which uses the computed impulse response as an input. (continued on back page)

DD FORM 1 JAN 73 1473

86 10 ... 097

DTIC FILE COPY

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. The research reported in this study seeks to define the limits and sensitivities of currently available deconvolution algorithms. In particular, the interrelationship among parameters of deconvolution procedures, noise, transducer bandwidth and instrumentation related errors were studied. Simulation experiments dealing with the impulse train recovery from material samples are illustrated in an algorithmic manner with the aid of a laboratory minicomputer system.

AFOSR-TR- 86-0904

FINAL TECHNICAL REPORT

August 15, 1984 to November 14, 1985

SIGNAL PROCESSING IN ULTRASONIC NDE

by

Pramode K. Bhagat, Ph.D.
Benjamin J. Leon, Ph.D.

University of Kentucky Research Foundation
University of Kentucky
Wenner-Gren Research Laboratory and
Department of Electrical Engineering
Lexington, KY 40506-0070

July 24, 1986

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

The authors acknowledge the assistance and support of the personnel both at the University of Kentucky and AFWAL/MLLP in carrying out the objectives of the study. In particular assistance of Tony Benge for software support, Greg Liming, Brad Bailey for automation of 3D positioning system and Anto Yauwanta in data analysis is acknowledged. The assistance of Dr. Moran and Mr. Shimmin in obtaining samples of flaw data is appreciated.

ABSTRACT

The desirable performance goal of an ultrasonic nondestructive evaluation, NDE, methodology is to reliably and rapidly obtain information regarding flaws in the material being tested. Decisions concerning acceptance/rejection of material for further usage can then be made on the basis of the nature and severity of flaws within it. Flaw size estimates are currently made on the basis of an idealized physical model, describing the flaw, which uses the computed impulse response as an input.

The research reported in this study seeks to define the limits and sensitivities of currently available deconvolution algorithms. In particular, the interrelationship among parameters of deconvolution procedures, noise, transducer bandwidth and instrumentation related errors were studied. Simulation experiments dealing with the impulse train recovery from material samples are illustrated in an algorithmic manner with the aid of a laboratory minicomputer system.

LIST OF FIGURES AND TABLES

INTRODUCTION

Ultrasound has been effectively used to characterize the internal structure of objects that are opaque to visible light. Historically the information contained in the ultrasonic signals, after passage through a test medium, was used to detect the presence of defects. Current research in ultrasonic testing towards a nondestructive evaluation capability requires extraction of information concerning size and shape of flaws from experimental data. This requires parametrization of the defect impulse response in terms of physical scattering mechanisms. While the impulse response of some geometrically well defined flaws (e.g. a spherical flaw) can be theoretically predicted, a theoretical frame work for arbitrary shaped flaws does not exist. Recent work using eigen-function expansion (also known as the singularity expansion method) for canonical objects such as spheres or spheriods suggests that size information can be extracted from the flaw impulse response in a pattern recognition approach. This approach is akin to assuming a transfer function model and extracting exponential terms from the computed impulse response. In both of these methods, described in the literature for defect sizing and classification, computation of impulse response becomes a necessary first step.

Modern signal processing techniques used in defining flaw impulse response are principally based on comparative analysis of a reference signal with the scattered flaw signal. The underlying hypothesis in these techniques is that, in the noise free case, the scattered flaw signal is due to the linear convolution of the ultrasonic reference signal with the flaw impulse response. The ultrasonic reference

signal is assumed to be a convolution of an electrical pulse with the instrumentation impulse response. The flaw characterization problem thus reduces to determining the kernel of the convolution integral given the input and output time signals. The impulse response recovery (system identification) has been carried out both in the frequency domain (Wiener filtering approach) and recently in time domain at AFWAL/MLLP (spline function approach). One of the authors, Dr. P. K. Bhagat, of this report has also implemented a nonlinear processing approach, namely homomorphic processing, at the Materials Laboratory (AFWAL/MLLP) for impulse recovery. This approach, utilizing logarithmic properties, is successful in identifying both the ultrasonic pulse used as well as the sample impulse response from simulated backscattered signal alone. Comparative analysis of these three techniques with synthetic data at AFWAL/MLLP reveals that, the recovered impulse responses are essentially equivalent in defining the time separation of reflectors, in the noise free case.

One theme that is common to the convolutional model is that the test sample is assumed to be a lossless medium, which is not realistic. Spectra of scattered signals from actual samples indicate that not only is the amplitude of the received signal decreased, but also that the center frequency is shifted. The shift in center frequency cannot be explained on the basis of a linear lumped parameter lossless model. This complicates the spectral division processing used to recover the flaw impulse response, since the dependence of the interrogating pulse on depth in the medium is not accounted for. Equivalently, in the time domain case, changes in the reference pulse shape are ignored.

It has been suggested that the test sample response be divided into several segments and the impulse response recovered on the basis that the interrogating pulse is invariant for the given segment. While this model is certainly more realistic, it compounds the impulse response recovery problem since ultrasonic pulses appearing in the intermediate segments cannot be defined experimentally.

A deconvolution procedure that can be applied to recovery of either of the two "short-time" convolved components when neither is known is homomorphic transformation. The assumption required under this approach is that the complex cepstra for impulse response and ultrasonic pulse do not overlap in the cepstral domain. The ultrasonic pulses used in NDE applications have a smooth spectrum and tend to occupy cepstral space around the origin whereas the flaw impulse response function appears as an impulse train. The spacing of the first impulse from cepstral origin, known as first arrival, determines whether the cepstra can be separated or not. If the first arrival and complex cepstrum of the ultrasonic pulse do not overlap then simple gating in the cepstral space can be used to recover both the impulse response and the ultrasonic pulse propagating through the medium.

The actual signals generated as a result of measurements in NDE applications are never noise free. A more realistic model would provide for convolutional component of noise due to coherent clutter and grain scattering plus an additive component due to electronic noise and random experimental errors. Since the time/frequency/ cepstral domain techniques are based on a linear convolutional relationship between the ultrasonic pulse and sample impulse response

straightforward application of these to system identification will lead to errors in estimating the actual impulse response. This problem is equivalently known as ill-posedness of the integral equation in the time domain case or ill-conditioning of spectral division in the frequency domain. In the cepstral domain the presence of convolutional and additive noise tends to blur the boundary between the pulse and impulse response cepstra leading to operator dependent recovery. Thus, in all the deconvolution procedures used, the impulse response recovery tends to be nonunique as the results tend to depend on the data processing requirements of data smoothing and filtering.

While there has been a major thrust in ultrasonic NDE research towards the development of inversion techniques, very little published literature exists on the dependence of the impulse response recovery on the signal processing procedures used. For example, the theoretical impulse response of a weakly scattering ellipsoidal object is well described in the literature. It is a function having a constant positive amplitude over the region corresponding to the interior of the object and two negative going peaks of short duration at each edge. This response has not been realized even for test samples containing spherical flaws and consequently the sizing algorithm, "inverse Born approach", has led to erroneous flaw size estimates. In addition, actual flaws may have rough surfaces which will tend to broaden the peaks in the recovered impulse response. The deconvolution procedure used, depending on the choice of algorithm parameters, will introduce additional broadening effects which will be data-dependent. Ever present noise in experimental data will further degrade the impulse response recovery process depending on the

particular algorithm used.

It is apparent from the foregoings that there is a need for development of a practical deconvolution procedure which provides impulse response recovery and identification based on a realistic model accounting for losses in the medium as well as the effects of noise. This study should be aimed at defining the limits and character of convolutional and additive noise components as well as segmentation of flaw signal for optimum impulse response recovery.

In an attempt to define the range of applicability and practical implementation of the current deconvolution procedures for NDE applications a simulation study was carried out as reported here. An integral feature of this study was the close collaboration between the principal investigator and the group led by Dr. T. J. Moran at AFWAL/MLLP in the development and implementation of signal processing techniques for impulse response recovery at their respective laboratories. The originally proposed three year study was, however, concluded after the first year when the principal investigator (P. K. Bhagat) accepted a new assignment outside the University.

The overall goal of the research study detailed herein was to establish both theoretically and in a practical setting the strengths and limitations of deconvolution procedures used in ultrasonic NDE with respect to noise characteristics and instrumentation employed. An integral feature of this study was the fact that the procedures are to be assessed using samples containing known flaws and samples where the flaw characteristics are unknown. We hope that the approach outlined here will lead to the development of a practical deconvolution procedure for routine field inspections.

BACKGROUND

The earliest approach to flaw characterization dates back to 1930 where the amplitude (intensity) of ultrasound was measured after its propagation through the material under test. Reduction in amplitude of received signal compared to a reference was interpreted as being caused by the flaw. Firestone (1,2) is credited as being the first to recognize the importance of the pulse echo method for application to nondestructive evaluation, NDE. The location of a flaw was defined through analysis of the received echo pattern; transit time measurement defines flaw location, amplitude of the echo is a function of flaw size, position and form of the flaw with respect to the transducer and the characteristics of the instrumentation used.

In general, flaw analysis approaches can be described as either parametric or nonparametric methods. By parametric methods, we mean those techniques which involve measurements of deterministic physical parameters resulting from the material-sound interaction equations. Velocity of sound propagation and attenuation are examples of these variables. Nonparametric methods involve essentially statistical measurement of variables which generally do not have well defined relationships in terms of physical phenomena. Feature extraction, using a pattern recognition approach, is an example of this methodology which has been used in the author's laboratory for characterization of tissue pathology (3).

Quantitative ultrasonic NDE techniques are based primarily on evaluation of impulse response (parametric approach) for definition of flaw location, shape and size. As has been described earlier the scattered flaw signal $s_f(t)$, is primarily a convolution of the

interrogating pulse, $s_r(t)$, with the flaw impulse response, $m_f(t)$

$$s_f(t) = s_r(t) * m_f(t) \qquad (1)$$

where * denotes convolution

Impulse response recovery from equation 1, known as deconvolution, has been addressed to by many authors and appears in several disciplines (for example see references 4-13). In the frequency domain the flaw transfer function, $M_f(jw)$, may be written as

$$M_f(jw) = S_f(jw) \frac{1}{S_r(jw)} \qquad (2)$$

where $S_f(jw)$ and $S_r(jw)$ are Fourier transforms of $s_f(t)$ and $s_r(t)$ respectively and w is the angular frequency.

As mentioned earlier, deconvolution has been carried out in the time/frequency/cepstral domain using interactive minicomputers. In the time domain, the impulse response recovery reduces to determining the kernel of an integral equation of the first kind. Phillips (5) has pointed out the ill-posed nature of this problem. Since the integral operator does not have a bounded inverse, a slight change in measured data will cause finite changes in the computed kernel values. Thus, in presence of noise, deconvolution using equation 1 will not provide a unique solution. Several authors (5,6) have developed computer based matrix algorithms which perform a given degree of smoothing on the data to minimize the effects of noise. Strand and Westwater (14)

have developed a general least square process of estimating the solution which also provides a measure of error in final results. Hunt (15) has developed a constrained deconvolution procedure using discrete Fourier transformation, which is equivalent to the works of Phillips (5) and Twomey (6). Recently Lee (16) has proposed a two stage solution procedure to the impulse recovery problem. In the first step the reference signal is fitted in the least square sense using spline functions. This fitted function is then used to provide a solution to the deconvolution problem. Thus the user can interactively define the degree of smoothing needed for his data analysis problem. The developed algorithm, due to the choice of spline function, is quite efficient in matrix manipulations and has been implemented on the AFWAL/MLLP computer (35).

In the frequency domain the problem of flaw transfer function recovery reduces to the design of an inverse filter. Presence of measurement noise ill-conditions the spectral division process as shown below:

$$M_f(jw) = \frac{1}{S_r(jw)} [S_f(jw) + N_2(jw)] \quad (3)$$

where $N_2(jw)$ is the noise spectra.

The ratio $N_2(jw)/S_r(jw)$ can completely dominate the $M_f(jw)$ computation in frequency bands of low SNR. In order to account for noise in the scattered signal Wiener filtering and constrained deconvolution have been used for the impulse recovery problem (10,26). Wiener filtering is based on the minimum integral mean square error whereas the constrained deconvolution uses a smoothness constraint function. Both approaches approximate equation 2 by:

$$M_f(jw) = \frac{S_r^*(jw)\ S_f(jw)}{S_r\ (jw)^2 + C^2} \qquad (4)$$

where C^2 is a user defined parameter and $S_r^*(jw)$ is complex conjugate of $S_r(jw)$

Since neither the properties of noise signals nor their energy content are known apriori, practical implementation in NDE application chooses C^2 so as to eliminate spectral division in low SNR regions. This approach has been used extensively in quantitative NDE work (27). Furgason et. al (26) have also reported on the deconvolution processing for flaw signatures and provide a constrained deconvolution filter which is based on earlier work of Hunt (15). These authors suggest that the deconvolution process can be made adaptive if the target response can be separated into known and unknown components. They provide impulse response computed from synthetic reflector series in support of their hypothesis.

Goebbels et. al (28) have considered the problem of grain scattering present in actual backscattered signals. They formulate grain scattering as a superposition of convolution outputs from each scatterer excited at the grain boundaries inside the sound beam and integrated over the path length. Their measurement model for scattered signal, x(t), can be described by:

$$x(t) = s_r(t) * [m_f(t) + n_1(t)] \qquad (5)$$

where $n_1(t)$ is the coherent noise component due to grains in the material sample.

These authors (28) minimize the grain scattering contributions through spatial averaging of the backscattered signal. This approach is based on the plausible hypothesis that small variations in the probe position cause significant changes in the grain scattered component but leave the component due to flaw essentially unaffected. Obviously this hypothesis implies that grain size is much smaller than reflector dimensions and ultrasonic wavelength used, and that the grains are randomly distributed in sample space.

In the cepstral domain, the defining equation corresponding to equation 2 is

$$F^{-1}[\log S_f(jw)]=F^{-1}[\log M_f(jw)]+F^{-1}[\log S_r(jw)] \quad (6)$$

where $F^{-1}[\cdot]$ is the inverse Fourier transform.

Under the assumption that the cepstra of the flaw impulse response and the interrogating pulse occupy disjoint spaces in the cepstral domain algorithms have been developed for deconvolution in speech processing, image processing, ultrasonic tissue characterization and seismic analysis. Recently the author has implemented a deconvolution procedure based on equation 6 on the AFWAL/MLLP computer (36). Simulation results obtained to date indicate that cepstral deconvolution procedure yields comparable results to the well established Wiener filter and time deconvolution procedure.

While limited literature exists on the topic of homomorphic processing applied to backscattered data analysis, there is a wealth of papers describing its application in other areas. Published

applications include removal of image blurs (17,18,19), speech processing (11,20), seisomology (7,21), and ECG signal analysis (22).

There have been several studies relating to the effects of additive noise on cepstral processing in the literature. Studies dealing with real cepstrum are due to Cole (17), Cannon (18), and Stockham (19). These authors were mainly interested in arriving at a power spectrum estimation of one of the convolved components such that a suitable gating filter could be implemented for deconvolution. Their methodology involved averaging the real cepstra of many segments of the convolved data contaminated with noise. They did not consider affects of noise on a single segment of data. Kemerait and Childers (23) have also studied the degradation caused by noise on the real cepstrum. They showed that real cepstral peaks, corresponding to reflector locations, are decreased in the presence of noise. The effect of additive noise on the complex cepstrum was less pronounced at SNR of 20 db. The work of Hassab et al (24) indicates that successful echo detection is dependent on the bandwidths of signal and noise. A study was recently carried out in the author's laboratory to investigate the effects of additive noise on recovery of convolutional components. An ultrasonic reference pulse was convolved with an arbitrary impulse train to produce a synthetic backscattered signal. Scaled Gaussian noise was added to the backscattered signal to produce the desired SNR in data. The complex cepstrum was computed using exponentially weighted data to remove the undesirable effects of spectral zeroes in unwrapping the phase. The results obtained, using symmetrical gating in the cepstral domain, indicate excellent recovery of the impulse train for SNR as low as 10 db (25). This author is unaware of any published work which takes into account the effect of

coherent noise on convolutional component recovery in cepstral processing.

A more realistic model of backscattered data from actual samples should contain contributions from both coherent and incoherent noise components. Such a model has been defined by Elsley et. al (27) who studied low frequency characteristics of flaws in ceramics. Their model, defining the received signal, y(t), in the time domain, is given by:

$$y(t) = s_r(t) * [m_f(t) + n_1(t)] + n_2(t) \qquad (7)$$

where $n_1(t)$ = noise component defining coherent clutter and grain scattering and,

$n_2(t)$ = Electronic random noise.

Considering each noise source separately with the scattering signal $s_f(t)$ these authors derive a constrained deconvolution filter. The form of filter (equation 4, this proposal) is essentially the same for either case, differing only in the choice of C^2. They conclude that choice of C^2 as a constant based on results obtained by their colleagues is satisfactory for deconvolution purposes. Elsley and Addison (29) have recently studied the accuracy of one dimensional Born estimation of flaw radius in the presence of noise. The synthetic waveforms used in this study were the calculated scattering from a spherical void to which scaled Gaussian noise was added. This noise was colored to accentuate Rayleigh or grain scattering [$(\text{frequency})^4$ dependence]. SNR was defined as the square root of the ratio of energy in the flaw signal to the mean energy in colored noise. Impulse response was computed from a constrained deconvolution

filter and integrated in the frequency domain to yield the characteristic function corresponding to the flaw. Radius of the flaw was then estimated by the ratio area/peak value of the characteristic function. For a flaw diameter of 800 μm a plot of estimated radius versus SNR is given. For zero dB SNR the estimate is shown to be off by ± 20%, however, one must note that this case implies very strongly scattering data from coherent noise sources. In actual practice preprocessing will reduce the extent of this noise. The characteristic function used in Born approximation has also been defined as an impediographic model [Jones (30)] or as a Raylograph [Beretsky and co-workers, (31, 32, 33)]. These authors compute the characteristic function as an integral of the calculated impulse response and also suggest an iterative procedure for optimization of amplitude and time location of the recovered impulse response. Their approach involves choosing a suitable band pass filter and combining its response with the spectra of the initial impulse response recovery to obtain a better estimate. While this analysis is shown in their paper to provide excellent results, this author is uncertain of the practical utility of this approach for quantitative NDE.

Bollig and Langenberg (34) have approached the ultrasonic defect classification problem in terms of transfer function modelling. These authors' attempt to peel off exponentials corresponding to singularities of the transfer function. These authors state "The crucial point in the SEM-parametrization of experimentally obtained time records of scattered ultrasonic signals is still the lack of an efficient algorithm to extract the singularities out of the transient data." They cite Prony's algorithm which has been used to extract the singularities from the transient response but state that the method is

quite sensitive to noise. These authors suggest that the singularities may be used in a feature extraction scheme from the system impulse response as a pattern recognition procedure for flaw sizing. They give results identifying creeping wave singularities and scattering cross section of spherical models for synthetic data.

MATERIAL AND METHODS

The experimental set up for data acquisition related to NDE tests is schematically shown in Figure 1. All the simulation experiments were conducted in a pulse echo made using a 5 MHz center frequency spherically focussed transducer. As shown in Figure 1 a PDP 11/23 minicomputer with 512 Kb memory and 479 Mb disk storage is the major element in the data acquisition procedure. All functions which include positioning of the transducer over material samples, energizing the transducer and capturing the digitized return echoes were carried out with the aid of the computer.

An MP 203 pulser (Metrotek, Inc) and an UTA-3 transducer analyzer (Aerotech Labs) were used to excite the transducer with an appropriate electrical pulse, gate and capture the reference and returning echoes from interfaces. The transducer, sample hold and the reflector were housed in a plexiglass tank filled with distilled water. In this study, both aluminum block and plexiglass reflectors were used.

Three dimensional movement of the ultrasonic transducer was facilitated through development of a positioning system. This system consists of three independent lead screw slide assemblies driven by stepper motors. Number of pulses input to the motors is controlled by user written software commands. Independent shaft encoders were implemented to verify and accurately position the transducers in a feedback arrangement. The current system provides for a normal incidence of the transducer on the sample with some degree of probe rotation (5 μm accuracy).

Signal digitization was carried out using a high speed A/D convertor (Biomation 8100) under the control of PDP 11/23. This A/D

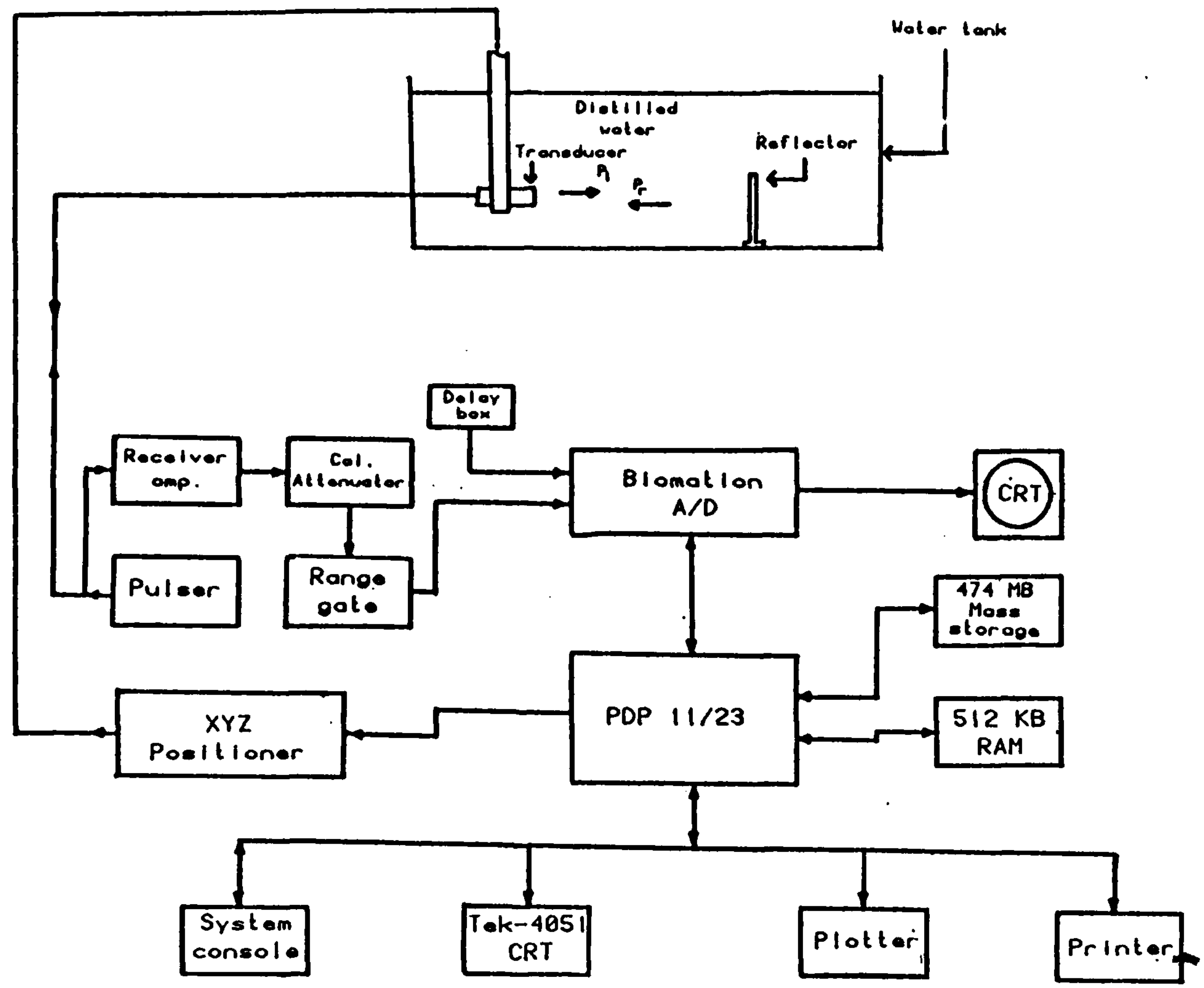


Fig.1 . Data acquisition and computer set up

convertor features a maximum sampling rate of 100 MHz and an interval memory of 2 K words. Under software control the digitized data was transferred to the mass storage unit of PDP 11/23. A CRT monitor (Tektronix 406) was provided to be able to view the digitized wave form as the data was being transferred.

Data analysis was performed on an offline basis using the PDP 11/23 computer. Software developed for the three deconvolution procedures (listed in Appendix) was used in user interactive fashion to generate simulation results. The programs and subroutines used in this study were generally adapted from literature to be compatible with PDP 11/23. In the following a brief summary of each procedure usage is given.

Time Domain Deconvolution (Spline fitting)

Figure 2 shows a block diagram of the Time Domain Deconvolution Process. As shown a user needs to specify several parameters prior to computation. These are: the knot ratio (KR), the order of solution spline (OS), and the number of basic spline function (#BSP). KR, once specified, is fixed for both data (the reference, $s(t_i)$, and the convolved, $y(t_i)$, data), while OS and #BSP can be varied independently by user.

For the data used in this experiment, KR values of 3 or 4, and OS values of 3 to 6 provide good fit. Maximum #BSP is 50 due to the limited memory space in the computer.

The computation of the basic spline functions are carried out in the subroutine BSP. The aim in fitting the reference data is to solve the linear equation Ac=b, by choosing c that minimizes the least

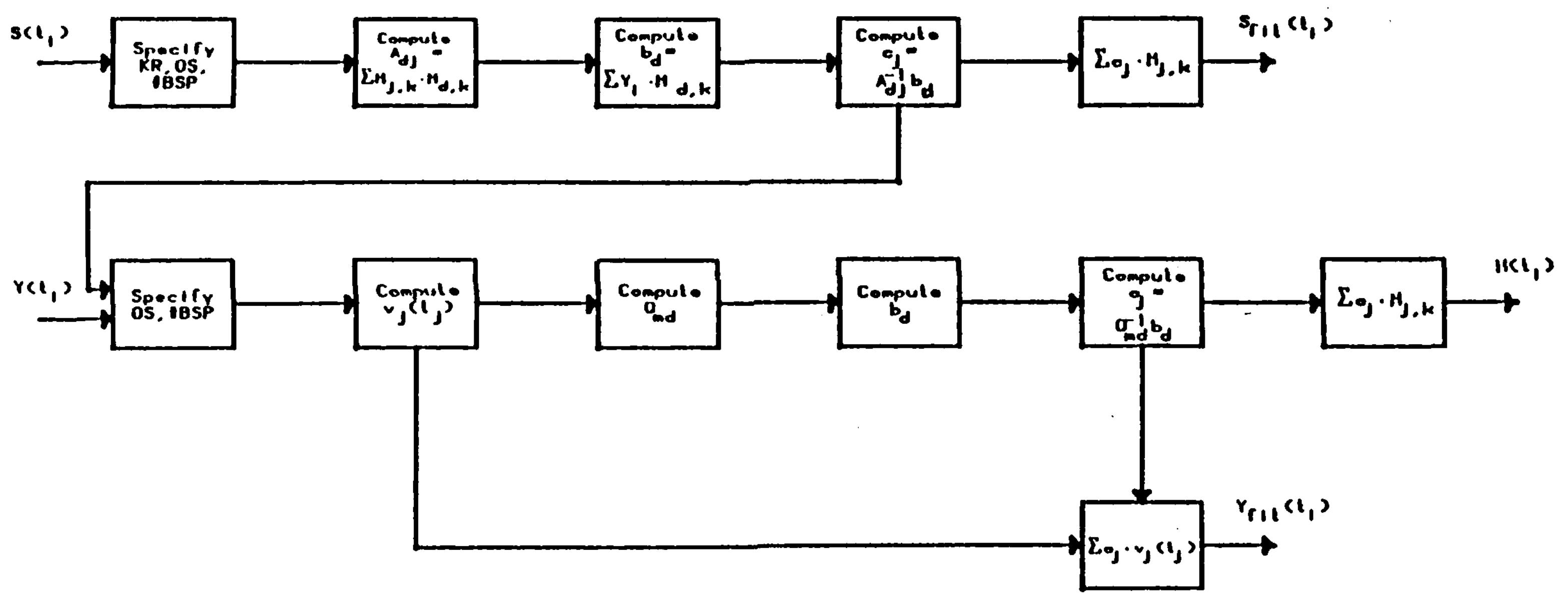


Fig. 2   Block diagram of the Time Domain Deconvolution process

square difference between the original and the smoothed reference data. Matrix inversion to compute $c = A^{-1}b$ is done in the subroutine BWS (the program listing is provided in the Appendix). Once the c_j are obtained, these values are then used to smooth the $y(t_i)$ data and extract the impulse response approximate solution, $h(t_i)$ from it.

Constrained Deconvolution (Wiener filtering)

The Constrained Deconvolution Process is shown in Figure 3. Utilizing the DFT routine, all the analysis is done in the frequency domain. The percent cutoff is a user supplied information, and is relative to the reference signal. It should be noted that in contrast to implementation of a statistical Wiener filter a preselected noise floor is used for deconvolution in this study (This is current practice in NDE).

By examining how much noise contaminated in the convolved data, a user specifies how much smoothing should be done. This smoothing is performed by setting the amplitude spectrum of the reference data, which has the value less than or equal to the relative percent cutoff to zero. Notice that by setting the amplitude spectrum to zero, the noise which usually occupy the low and high frequency will be leveled off.

Homomorphic Deconvolution

Figure 4 shows The Homomorphic Deconvolution Process schematically. Some important points to be considered in this process are given below.

1. Append zeros

Appending zeros provides two advantages. By increasing

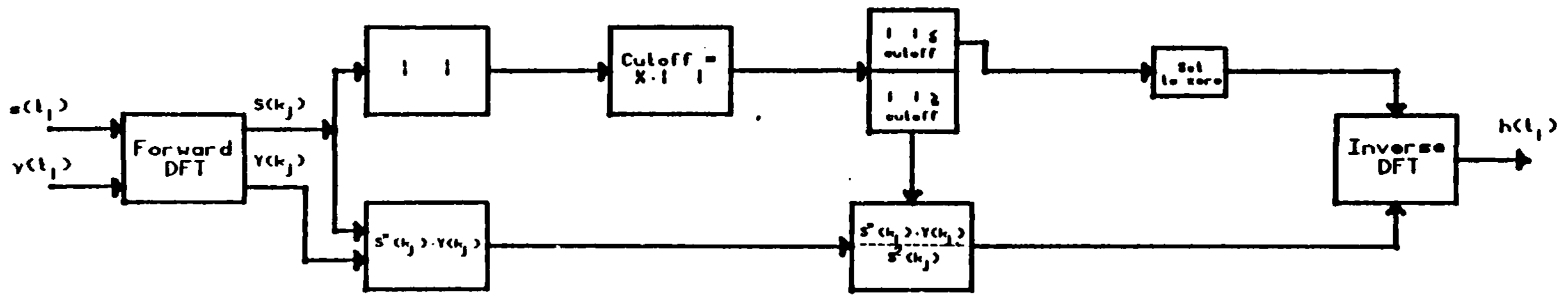


Fig. 3 Block diagram of the Constrained Deconvolution process

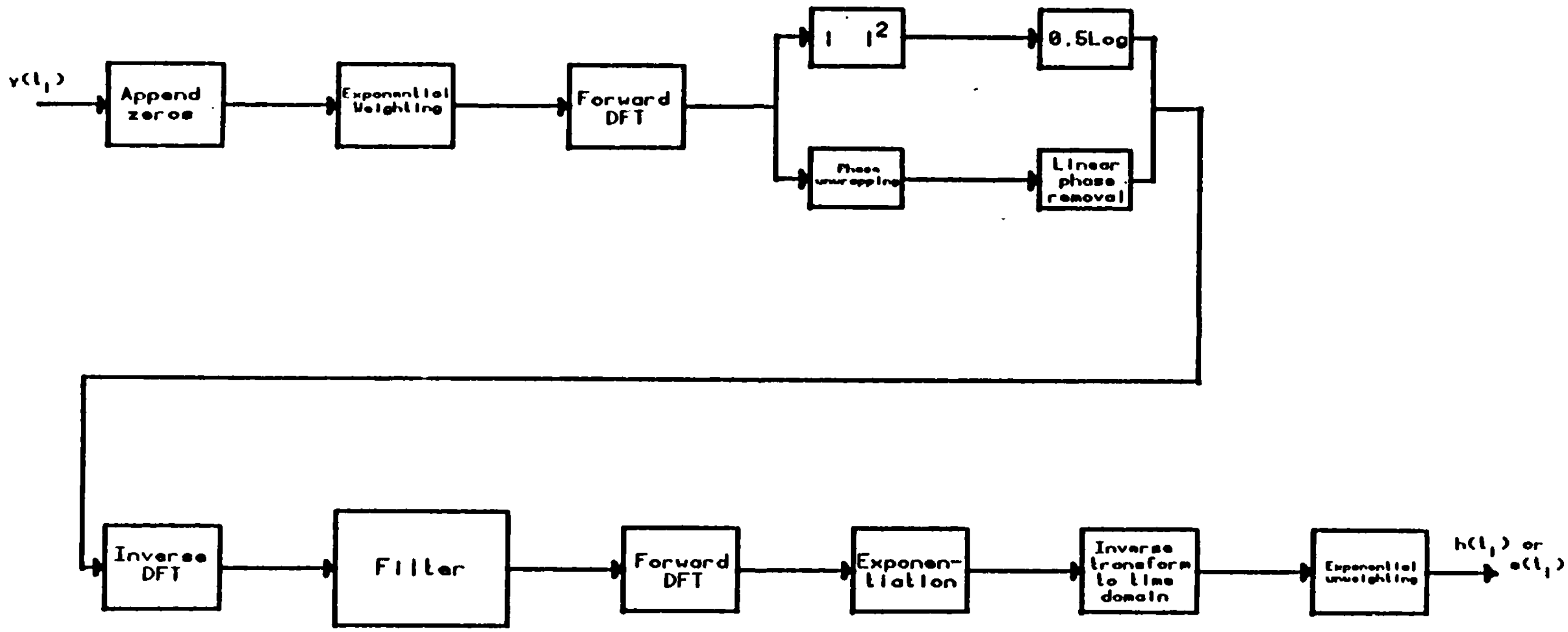


Fig. 4 Block diagram of the Homomorphic Deconvolution process

the number of samples aliasing due to computation logarithm and phase unwrapping errors caused by linear phase components are reduced.

Exponential Weighting

The method of exponential weighting was first introduced by Schafer, discussed in [21], to convert a mixed phase sequence into a minimum phase sequence. This is accomplished through multiplication of input data sequence by a real number a^n, where $0<a<1$ and n is the data sequence number. Choice of the weighting, a, is data dependent.

Phase Unwrapping

Since the arctangent routine in the computer only evaluates the phase of modulo 2π, discontinuities occur in the phase curve. The computation of the logarithm of the data requires that the phase must be analytic in some annular region of the z-plane [21]. Therefore, phase unwrapping is necessary to avoid these discontinuities. By adding the appropriate multiple of 2π to the principal values of the phase, unwrapping is achieved.

Filtering

Linear filtering is applied in the complex cepstrum to decompose the convolved sequences. This is done by setting the complex cepstrum due to the contribution of one or the other signal to zero. Two type of filters have been used, they are low-pass and comb or notch filter. Low-pass filtering is used

when the first arrival is easily detected, and is done by setting the complex cepstrum to zero beyond the first arrival. The comb filtering is used when the complex cepstrum is badly corrupted by noise that the first arrival cannot be detected.

The Inverse Process

Inverse processing performed to transform the complex cepstrum into its time domain. Reinsertion of the linear phase is done in this process to obtain the original phase from its shifted version caused by the linear phase removal. Exponential unweighting is applied to the sequences to restore effect of the exponential weighting done in the beginning of the process.

Noise and DFT Routines

In order to make the data more realistic, simulated random noise is added to the convolved data. The random noise is generated by the program ADDNOI, by specifying the statistical level of the noise to the data.

Simulation Experiments

1. Noise Free Case

Front surface echo was used as the reference signal. This signal was convolved with arbitrary impulse trains to generate simulated flaw data. The polarity and relative amplitudes of the impulses were defined to portray backscattered signals from inclusions or voids. The separation between impulses was varied according to the ultrasonic wavelength used in an effort to

define resolution of the techniques used.

The synthesized signa_, $S_f(t)$, was deconvolved using developed programs for constrained deconvolution, cepstral processing the spline function deconvolution. A figure of merit, η, defined as:

$$\eta \triangleq \int_0^T [\tilde{m}_f(t_i) - m_f(t_i)]^2 dt$$

where $\tilde{m}_f(t_i)$ is the recovered impulse

was used to characterize the success/failure of each deconvolution technique used. The parameters of the deconvolution procedure used were varied to obtain optimum results. This implies varying the knot-spacing, order of spline and number of splines in case of time domain deconvolution; transducer cut-off frequency, smoothing operator and size of FFT in case of constrained deconvolution (also called Wiener filter in ultrasonic NDE) and exponential weighting, FFT size and linear filter/gate employed in cepstral processing.

In addition, the impulse response recovery problem was studied with impulses lying within an ultrasonic wavelength of each other in order to define the limitation of the deconvolution procedures in identifying closely spaced reflectors.

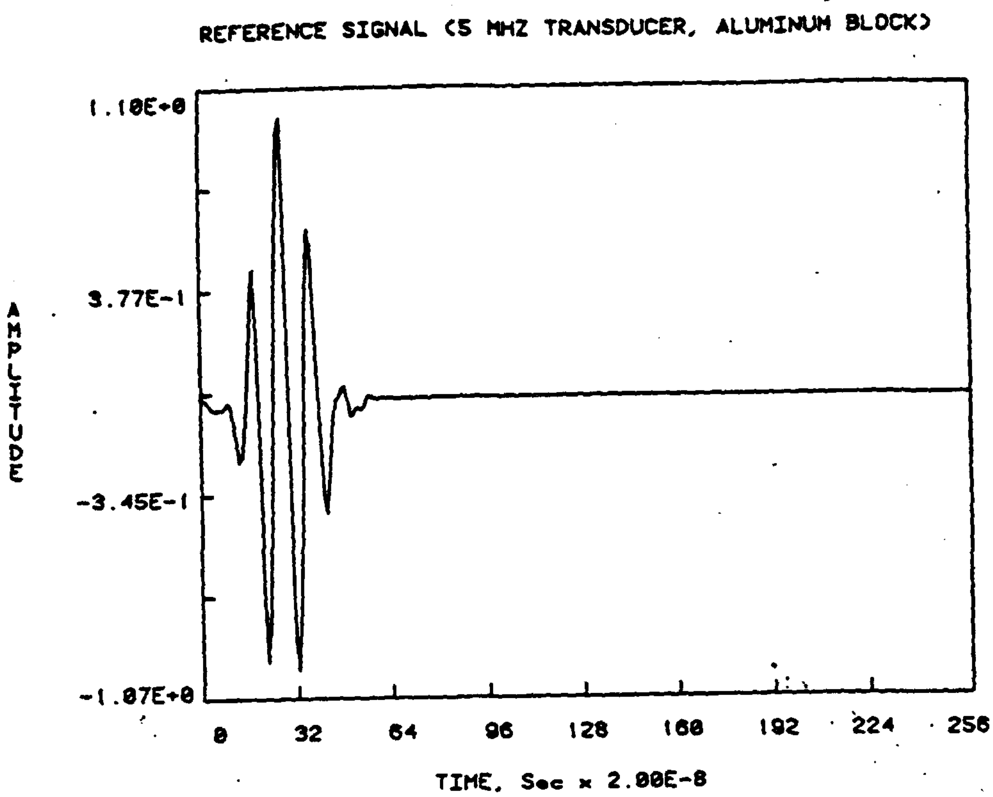
2. Coherent Noise Case

Coherent noise was generated using a standard Gaussian noise algorithm already in place on our computer. This noise in some cases will be colored by $(\text{frequency})^4$ weighting to simulate Rayleigh scattering. The impulse response recovery as a function

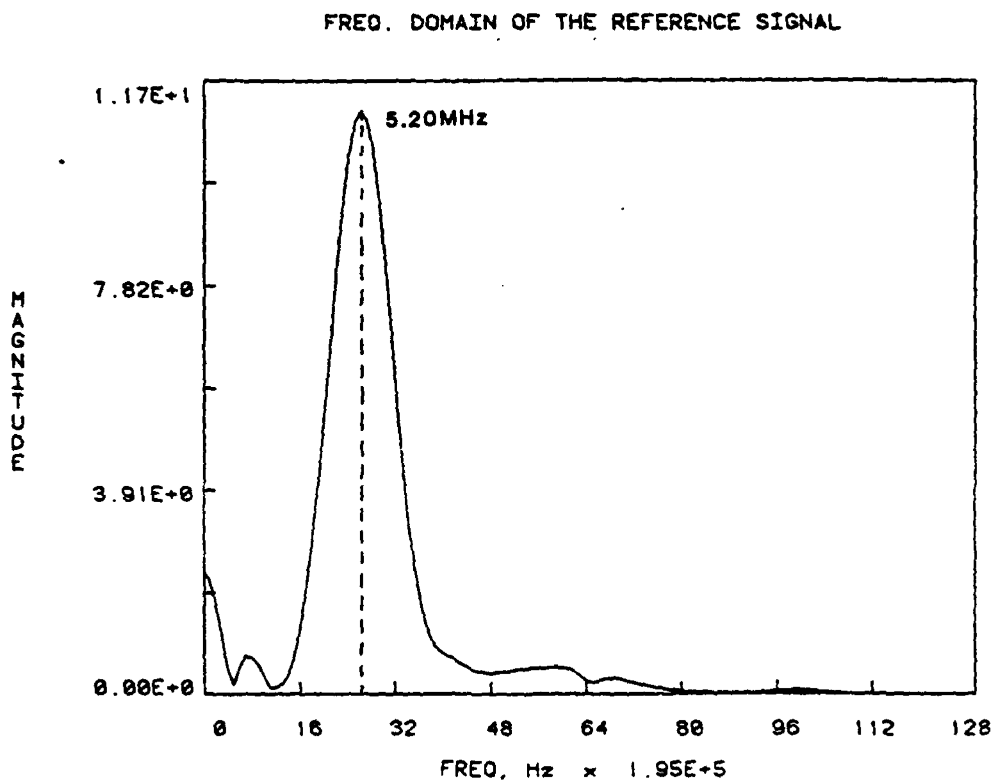
of noise level was studied. Limited experiments have already indicated that good impulse response recovery is possible for SNR as low as 6 db. However, these experiments were carried out with wider separation between impulses; and the interactive aspects of both noise and pulse separation were not fully explored.

RESULTS AND DISCUSSION

Figure 5 shows the time domain plot of the ultrasonic reference signal used in the study. This signal was obtained as the front surface echo, digitized at 50 MHz, from an aluminum block reflector. The transducer nominal frequency was 5 MHz. As can be seen the actual transducer frequency is 5.20 MHz with significant high frequency response up to 10 MHz. Figure 6 shows the complex cepstrum of data obtained as a result of convolving the reference signal with varying interface (impulse) separations. At an interface separation of 25 λ (wavelength), the first arrival (seen as a small sharp peak) can be easily discerned. In contrast when the pulse separation was reduced to 1.5 λ, the first arrival is submerged in the cepstrum of the reference signal. Signal separation in such situations is difficult. Impulse response recovery as a function of exponential weight is shown in Figure 7. The simulated signal represents a spherical flaw within a material sample. Increasing the exponential weight tends to smoothen the recovered impulse response. The presence of high frequency data in the results is due mainly to inverse logarithm (exponentiation) function. Also note the displacement of time origin of data due to linear phase (time delay). Figure 8 shows impulse response recoveries for the three deconvolution methods KR=3 and OS=4 shows good recovery but there are a lot of oscillations in the data. In the absence of 'apriori' knowledge of interface separations it would be difficult to identify the structures in the recovered impulse response. Constrained deconvolution (Wiener filter) using Fourier transformed reference and convolved data provides excellent recovery as indicated. Homomorphic deconvolution with gate width of 20 samples



(a)



(b)

Fig. 5. Reference signal obtained through aluminum block reflector.
a. in time domain b. in frequency domain

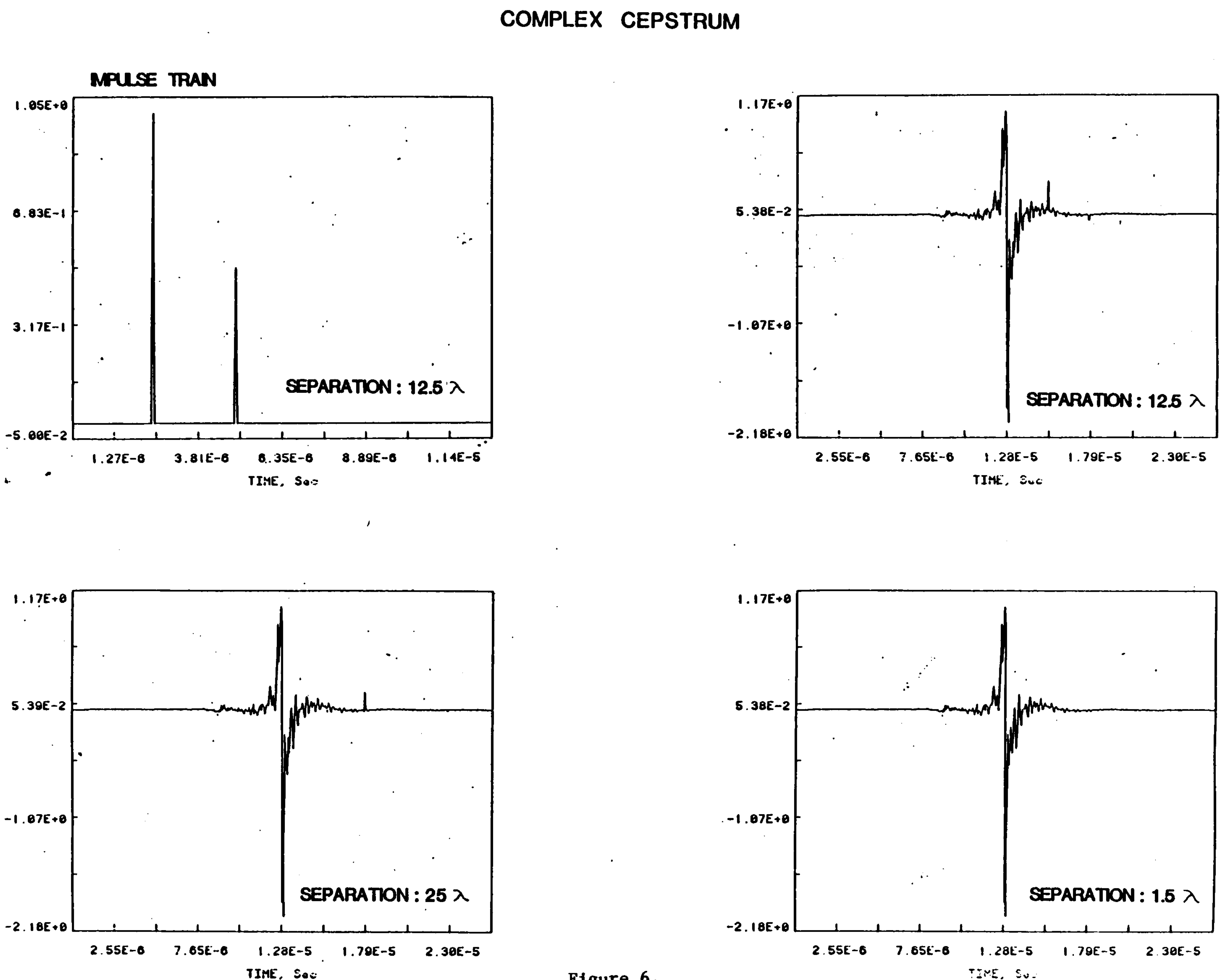

COMPLEX CEPSTRUM
IMPULSE TRAIN
1.05E+0
6.83E-1
3.17E-1
-5.00E-2
1.27E-6
3.81E-6
6.35E-6
8.89E-6
1.14E-5
TIME, Sec
SEPARATION : 12.5 λ
1.17E+0
5.38E-2
-1.07E+0
-2.18E+0
2.55E-6
7.65E-6
1.28E-5
1.79E-5
2.30E-5
SEPARATION : 12.5 λ
5.39E-2
SEPARATION : 25 λ
SEPARATION : 1.5 λ

Figure 6.

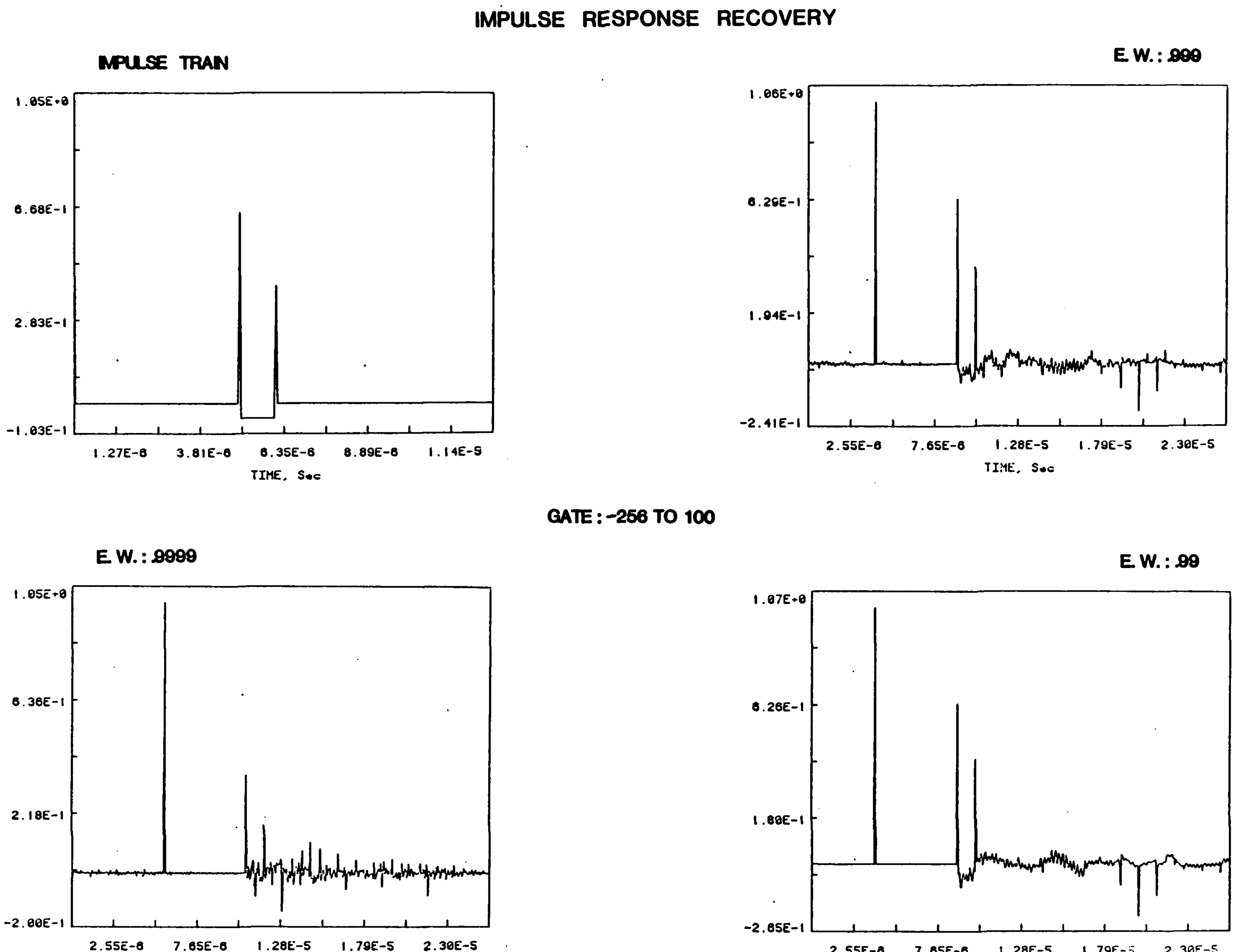

IMPULSE RESPONSE RECOVERY
IMPULSE TRAIN
1.05E+0
6.68E-1
2.83E-1
-1.03E-1
1.27E-6
3.81E-6
6.35E-6
8.89E-6
1.14E-5
TIME, Sec
E.W.: .999
1.06E+0
6.29E-1
1.94E-1
-2.41E-1
2.55E-6
7.65E-6
1.28E-5
1.79E-5
2.30E-5
TIME, Sec
GATE: -256 TO 100
E.W.: .9999
1.05E+0
6.36E-1
2.18E-1
-2.00E-1
2.55E-6
7.65E-6
1.28E-5
1.79E-5
2.30E-5
E.W.: .99
1.07E+0
6.26E-1
1.80E-1
-2.65E-1
2.55E-6
7.65E-6
1.28E-5
2.30E-5

Figure 7.

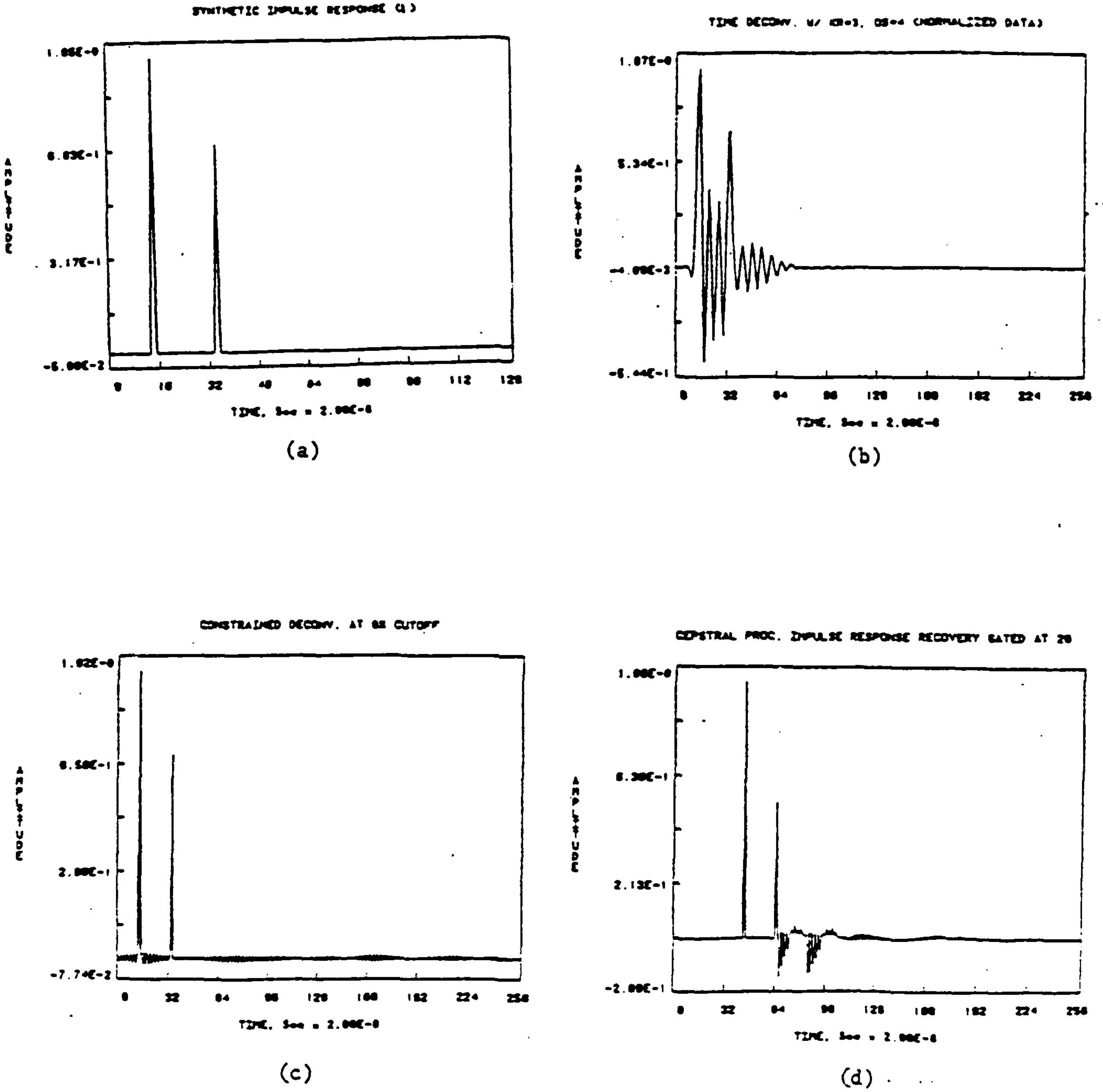


Fig. 8 Impulse response recovery, no-noise case.
a. synthetic impulse response
b. TDDM recovery
c. CDM recovery
d. HDM recovery

and exponential weight of 0.9999 also provides good recovery. The presence of small amplitude high frequency oscillations is a cause for concern. At a SNR of 30 dB the impulse response recovery is degraded as shown in Figure 9. Note the smoothing function of the time domain deconvolution. There is no perceptible change in the shape of the recovered impulse train. In contrast, both the Wiener and cepstral processed data show significant noise floor in the recovered impulse train. Figure 10 and 11 show the impulse recoveries for 20 dB and 10 dB SNR. Degradation of impulse response recovery is apparent in all the methods. Interestingly, at 10 dB SNR time domain deconvolution provides sharpened peaks coupled with low frequency interference. In contrast, homomorphic deconvolution shows large amplitude noise in the recovered data. Table 1 shows the normalized RMS error in recovering the impulse train. Figure 12 shows the percentage deviation of Born estimate of flaw dimensions based on recovered impulse responses. Degraded performance is apparent in all the three deconvolution procedures used.

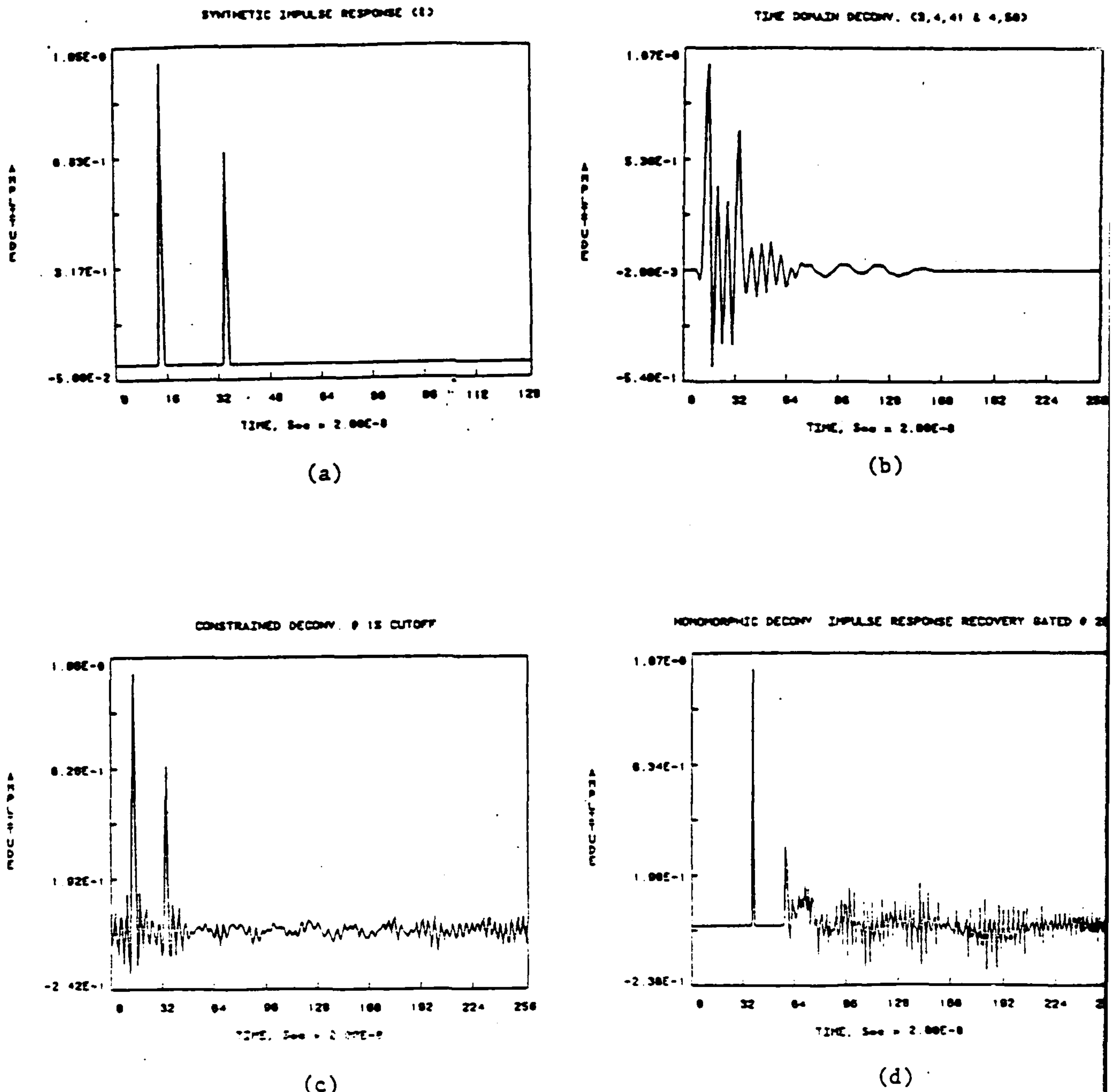


Fig. 9 Impulse Response recovery, + 30dB SNR case.
a. synthetic impulse response
b. TDDM recovery
c. CDM recovery
d. HDM recovery

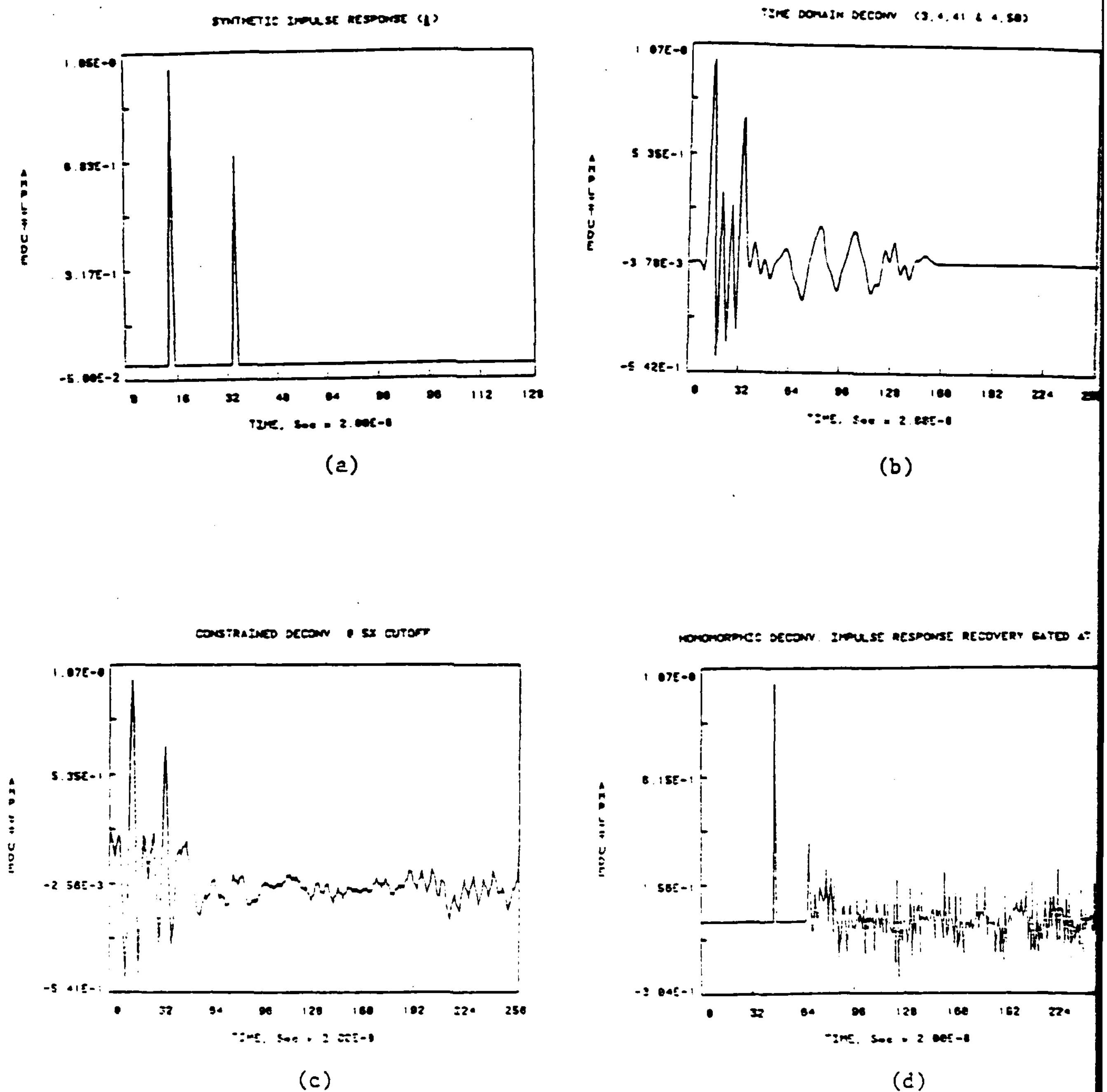


Fig. 10 Impulse response recovery, + 20dB SNR case.
a. synthetic impulse response
b. TDDM recovery
c. CDM recovery
d. HDM recovery

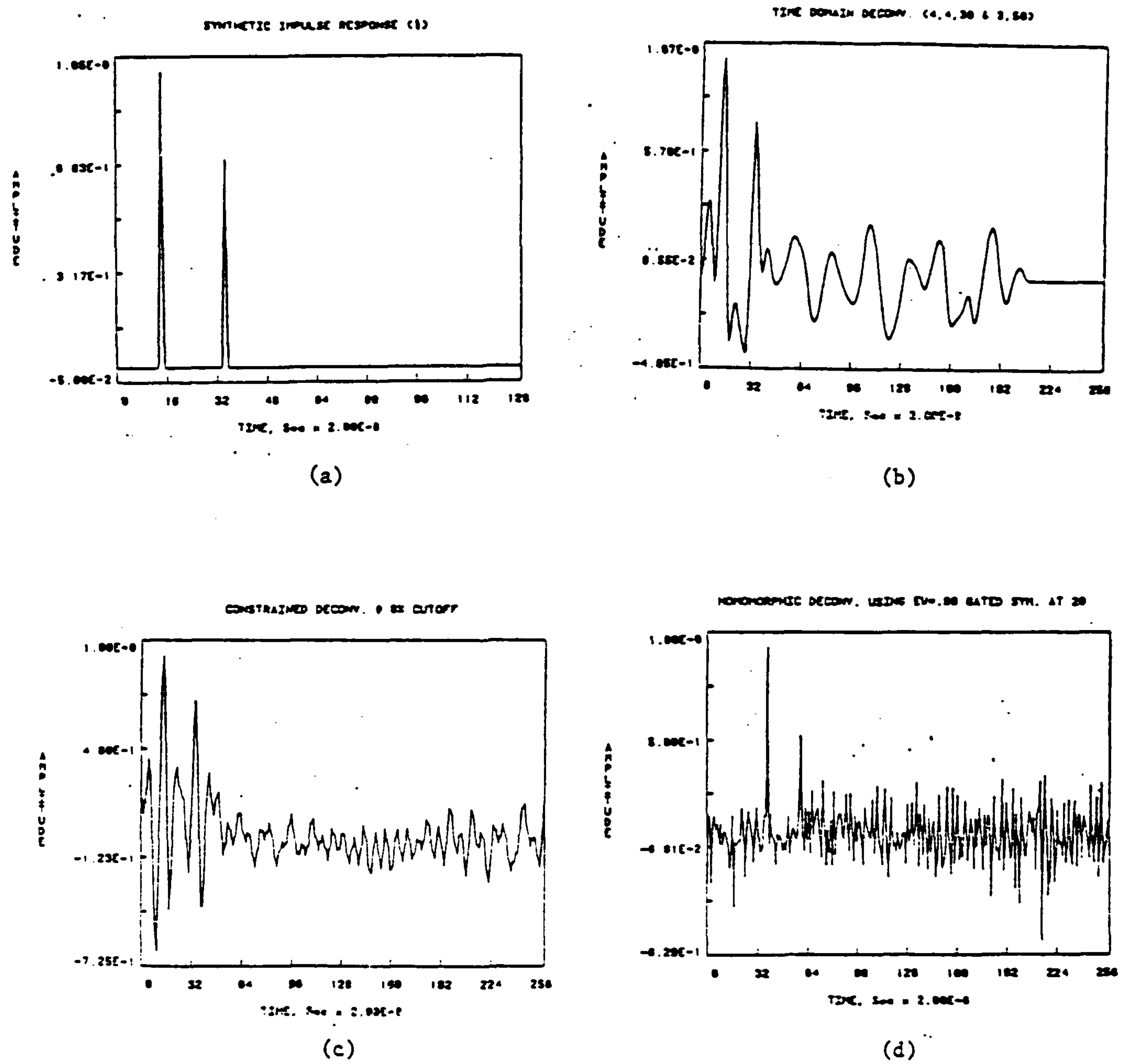


Fig. 11 Impulse response recovery, + 10dB SNR case.
a. synthetic impulse response
b. TDDM recovery
c. CDM recovery
d. HDM recovery

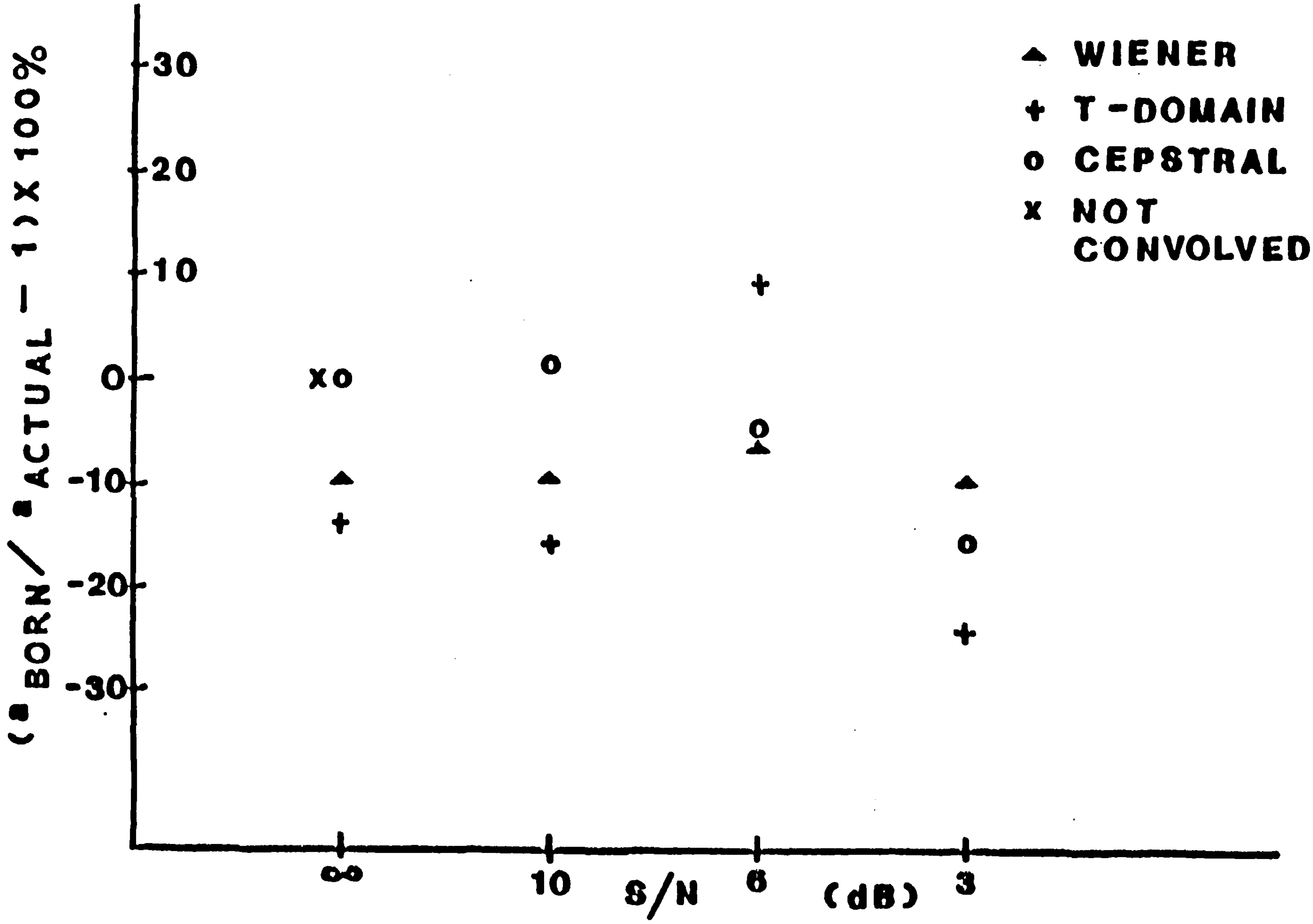


FIGURE 12. % Deviation of Born Estimate Vs. S/N Ratio

Normalized RMS Error				
Method of Deconvolution	Noise Level			
	No-noise	+30dB SNR	+20dB SNR	+10dB SNR
Time Domain	15.63%	15.76%	17.00%	20.30%
Constrained	1.35%	9.94%	16.93%	20.72%
Homomorphic	14.17%	32.58%	30.66%	46.56%

Table 1. Normalized Root Mean Square Error in recovering the Impulse Response. Analyzed for data obtained through Aluminum block reflector (2 Impulses)

SUMMARY

An NDE test setup with capability of computer based ultrasonic flaw signal analysis is detailed in this study. Extensive software development along with fabrication of a three dimensional positioning system formed a major portion of the study. Limited simulation results indicate that the deconvolution procedures are greatly dependent on the SNR of data. Care must be exercised in using the deconvolved data to estimate flaw dimensions due to the presence of undesirable noise in the recovered impulse response. Other approaches such as autoregressive modeling of the signal might provide a vehicle for noise suppression and smoothing.

SIGNIFICANCE OF RESEARCH

Careful documentation of sensitivities and limitations of deconvolution procedures for impulse response recovery will constitute a major contribution to the field of quantitative ultrasonic NDE. Since all the physical models developed to data for flaw sizing require computation of sample transfer function as a necessary first step, the research outlined here will provide both a theoretical basis as well a practical limitations on impulse recovery process. If we are successful in developing a practical deconvolution procedure for real time application under actual field conditions, significant improvement in flaw sizing capability will result. The interdisciplinary nature of the project and collaboration with Air Force personnel should serve to promote NDE research directly oriented towards Air Force needs and increase University participation in this kind of activity.

REFERENCES CITED

1. Firestone, F. A., Supersonic Reflectoscope, an Instrument for Inspecting the Interior of Solid Parts by Means of Sound Waves, J. Acoust. Soc. Amer., 17, 287-299, 1945.

2. Firestone, F. A., Tricks With the Supersonic Reflectoscope, Nondest. Test, 7, 5-19, 1948.

3. Dyer, R. A. G., Classification and Characterization of Tissue Pathology Through Ultrasonic Signal Analysis, Ph D Dissertation, University of Kentucky, Lexington, KY, 1980.

4. Eykhoff, P., System Identification: Parameter and State Identification, Wiley, New York, 1974.

5. Phillips, D. L., A Technique for the Numerical Solution of Certain Integral Equations of the First Kind, J. Assoc. Comput. Mach., 9, 84-97, 1962.

6. Twomey, S., The Application of Numerical Filtering to the Solution of Integral Equations Encountered in Indirect Sensing Measurements, J. Franklin Inst., 279, 95-109, 1965.

7. Ulrych, T. J., Application of Homomorphic Deconvolution of Seismology, Geophysics, 36, 650-660, 1971.

8. Hunt, B. R., Digital Image Processing, Proc. IEEE (special issue on Digital Signal Processing), 63, 693-708, 1975.

9. Oppenheim, A. V., Shafer, R. W., and Stockham, T. G., Jr., Nonlinear Filtering of Multiplied and Convolved Signals, Proc. IEEE, 56, 1264-1291, 1968.

10. Kak, A. C. and Dines, K. A., Signal Processing of Broadband Pulsed Ultrasound: Measurement of Attenuation of Soft Biological Tissues, IEEE Trans. BioMed. Engin., BME-25, 321-344, 1978.

11. Oppenheim, A. V., Speech Analysis-Synthesis System Based on Homomorphic Filtering, J. Acoust. Soc. Amer., 45, 459-462, 1969.

12. Makhoul, J., Linear Prediction: A Tutorial Review, Proc. IEEE, 63, 561-580, 1975.

13. Murakami, Y., Khuri-Yakub, B. T., Kino, G. S., Richardson, J. M., and Evans, A. G., An Application of Wiener Filtering to Nondestructive Evaluation, Appl Phys Letters, 33, 685-687, 1978.

14. Strand, O. and Westwater, E., Statistical Estimation of the Numerical Solution of a Fredholm Integral Equation of the First Kind, J. Assoc. Comp. Mach., 15, 100-114, 1968.

15. Hunt, B. R., The Inverse Problem of Radiography, Math Biosci., 8, 161-179, 1970.

16. Lee, D. A., Scatterer Sizing from Elastodynamic Backscattering Using Spline, Proceedings of the 12th Annual Pittsburgh Conference, Modelling and Simulation, Vol. 12 (4), 1253-1257, 1981.

17. Cannon, M., Blind Deconvolution of Spatially Invariant Image Blurs With Phase, IEEE Trans. Acoust. Sp. Sig. Proc., ASSP-24, 58-63, 1976.

18. Cole, E. R., The Removal of Unknown Image Blurs by Homomorphic Filtering, Ph.D. Dissertation, University of Utah, 1973.

19. Stockham, T. G., Jr., Cannon, T. M., and Ingebretsen, R. B., Blind Deconvolution Through Digital Signal Processing, Proc IEEE, 63, 678-692, 1975.

20. Oppenheim, A. V. and Schafer, R. W., Homomorphic Analysis of Speech, IEEE Trans. Audio Electroacoust., AV-16, 221-226, 1968.

21. Tribolet, J. M., Seismic Application of Homomorphic Signal Processing, Prentice-Hall, New Jersey, 1979.

22. Murthy, I. S. N., Rangaraj, M. R., Udupa, K. J., and Goyal, A. K., Homomorphic Analysis and Modeling of ECG Signals, IEEE Trans. Bio-Med Engin., BME-26, 335-344, 1979.

23. Kemerait, R. C. and Childers, D. G., Signal Detection and Extraction by Cepstrum Techniques, IEEE Trans. Info. Theory IT-18, 745-759, 1972.

24. Hassab, J. C. and Boucher, R., A Probabilistic Analysis of Time Delay Extraction by the Cepstrum in Stationary Gaussian Noise, IEEE Trans. Info. Theory, IT-22, 444-454, 1976.

25. Shafer, M. E., The Application of Homomorphic Processing to Ultrasonic Signals, M.S. Thesis, University of Kentucky, Lexington, KY, 1981.

26. Furgason, E. S., Twyman, R. E., and Newhouse, V. L., Deconvolution Processing for Flaw Signatures, 312-318, Proceedings ARPA/AFML Review of Progress in Quantitative NDE, May 1978.

27. Elsley, R. K., Ahlbert, L. A., Richardson, J. M., Low Frequency Characterization of Flaws in Ceramics, 151-163, Proceedings DARPA/AFWAL Review of Progress in Quantitative NDE, AFWAL-TR-81-4080, 1981.

28. Goebbels, K., Kraus, S., and Neumann, R., Fast Signal Averaging Unit for Ultrasonic Testing. Characterization of Material Properties and SNR - Improvement for Coarse Grained Materials, 437-444, Proceedings DARPA/AFML Review of Progress in Quantitative NDE, AFWAL-TR-80-4078, 1980.

29. Elsley, R. K., and Addison, R. C., Dependence of the Accuracy of the Born Inversion of Noise and Bandwidth, 389-395, Proceedings DARPA/AFWAL Review of Progress in Quantitative NDE, AFWAL-TR-81-4080, 1981.

30. Jones, J. P., Impediography: A New Ultrasonic Technique for Diagnostic Medicine, Ultrasound in Medicine, D. White (Ed), Volume 1, 489-497, Plenum, 1976.

31. Beretsky, I., Raylography: A Frequency Domain Processing Technique for Pulse Echo Ultrasonography, Ultrasound in Medicine, D. White (Ed), Volume 3B, 1581-1596, Plenum, 1976.

32. Papoulis, A., and Beretsky, I., Improvement of Range Resolution by Spectral Extrapolation, Ibid, 1613-1627.

33. Beretsky, I., and Farrell, G. A., Improvement of Ultrasonic Imaging and Media Characterization by Frequency Domain Deconvolution: Experimental Study with Nonbiological Material, Ibid, 1645-1665.

34. Bollig, G., and Langenberg, K. J., Ultrasonic Defect Classification Using the Singularity Expansion Method, 203-212, Review of Progress in Quantitative Nondestruction Evaluation, D. Thompson and D. Chimenti (Eds), Plenum, 1982.

35. Bhagat, P. K., Application of Advanced Signal Processing Methods to NDE Problems, Quarterly Progress Report, Submitted to AFWAL/MLLP, March 1983.

36. Bhagat, P. K. and Shimmin, K., Homomorphic Processing in Ultrasonic NDE. Presented at the Eighth DARPA/AFWAL Review of Progress in Quantitative NDE, Santa Cruz, California, August 1983.

37. Kadaba, M. P., Bhagat, P. K., and Wu, V. C., Attenuation and Backscattering of Ultrasound in Freshly Excised Animal Tissue, IEEE Trans. Biomed. Engin., BME-27, 76-83, 1980.

APPENDIX

Software Listings

Complete FORTRAN listings of the deconvolution procedures used in this study are provided. The programs are generally in subroutine form and therefore can be adapted to different computer systems with a minimal amount of modifications.

```
      SUBROUTINE CCEPS(NX,X,ISNX,ISFX,ISSUC,CX,AUX)
C
C     The subroutines used in this program were taken from:
C
C     'Programs for Digital Signal Processing'
C     IEEE Press, 1979
C     345 East 47 Strett, New York, NY 10017
C     Sponsored by the IEEE Acoustics, Speech, and
C                           Signal Processing Society
C     Lib. of Congress Cat. Card # 79-89028
C     IEEE Book # 0-87942-128-2 (paperback ver.)
C                # 0-87942-127-4 (hardback)
C     Also published by John Wiley & Sons, Inc.
C     Wiley Order # 0-471-05961-7 (paperback ver.)
C                   # 0-471-05962-5 (hardback)
C
      DIMENSION X(1),CX(1),AUX(1)
      COMMON PI,TWOPI,THLINC,THLCON,NFFT,NPTS,N,L,H,H1,DVTMN2
      LOGICAL ISSUC
C
C     SUBROUTINE TO COMPUTE THE COMPLEX CEPSTRUM
C
      NPTS=NFFT/2
      N=12
      L=2**N
      H=FLOAT(L)*FLOAT(NFFT)
      H1=PI/H
      ISSUC=.TRUE.
      ISNX=1
C  TRANSFORM X(N) AND N*X(N)
C
      DO 10 I=1,NX
      CX(I)=X(I)
      AUX(I)=FLOAT(I-1)*X(I)
10    CONTINUE
C APPEND THE NECESSARY ZEROES FOR FFT COMPUTATION
C
      INITL=NX+1
      IEND=NFFT+2
      DO 20 I=INITL,IEND
      CX(I)=0.0
      AUX(I)=0.0
20    CONTINUE
C
C COMPUTE FFT
      CALL FAST(CX,NFFT)
      CALL FAST(AUX,NFFT)
C CHECK IF SIGN REVERSAL IS REQUIRED
C
      IF(CX(1).LT.0.0)ISNX=-1
C
C
C
C  COMPUTE MAGNITUDE OF SPECTRUM ;STORE IN ODD INDEXED VALUES OF
C  AUX
C
C COMPUTE PHASE DERIVATIVE OF THE SPECTRUM : STORE IN EVEN INDEXED
C VALUES OF AUX
C
      IO=-1
```

```
      SUBROUTINE CCEPS(NX,X,ISNX,ISFX,ISSUC,CX,AUX)
C
C     The subroutines used in this program were taken from:
C
C     'Programs for Digital Signal Processing'
C     IEEE Press, 1979
C     345 East 47 Strett, New York, NY 10017
C     Sponsored by the IEEE Acoustics, Speech, and
C                     Signal Processing Society
C     Lib. of Congress Cat. Card # 79-89028
C     IEEE Book # 0-87942-128-2 (paperback ver.)
C               # 0-87942-127-4 (hardback)
C     Also published by John Wiley & Sons, Inc.
C     Wiley Order # 0-471-05961-7 (paperback ver.)
C                 # 0-471-05962-5 (hardback)
C
      DIMENSION X(1),CX(1),AUX(1)
      COMMON PI,TWOPI,THLINC,THLCON,NFFT,NPTS,N,L,H,H1,DVTMN2
      LOGICAL ISSUC
C
C     SUBROUTINE TO COMPUTE THE COMPLEX CEPSTRUM
C
      NPTS=NFFT/2
      N=12
      L=2**N
      H=FLOAT(L)*FLOAT(NFFT)
      H1=PI/H
      ISSUC=.TRUE.
      ISNX=1
C  TRANSFORM X(N) AND N*X(N)
C
      DO 10 I=1,NX
      CX(I)=X(I)
      AUX(I)=FLOAT(I-1)*X(I)
10    CONTINUE
C APPEND THE NECESSARY ZEROES FOR FFT COMPUTATION
C
      INITL=NX+1
      IEND=NFFT+2
      DO 20 I=INITL,IEND
      CX(I)=0.0
      AUX(I)=0.0
20    CONTINUE
C
C COMPUTE FFT
      CALL FAST(CX,NFFT)
      CALL FAST(AUX,NFFT)
C CHECK IF SIGN REVERSAL IS REQUIRED
C
      IF(CX(1).LT.0.0)ISNX=-1
C
C
C
C  COMPUTE MAGNITUDE OF SPECTRUM :STORE IN ODD INDEXED VALUES OF
C  AUX
C
C COMPUTE PHASE DERIVATIVE OF THE SPECTRUM : STORE IN EVEN INDEXED
C VALUES OF AUX
C
      IO=-1
```

```
      DVTMN2=0.0
      IEND=NPTS+1
      DO 30 I=1,IEND
      IO=IO+2
      IE=IO+1
      AMAGSQ=AMODSQ(CX(IO),CX(IE))
      PDVT=PHADVT(CX(IO),CX(IE),AUX(IO),AUX(IE),AMAGSQ)
      AUX(IO)=AMAGSQ
      AUX(IE)=PDVT
      DVTMN2=DVTMN2+PDVT
30    CONTINUE
      DVTMN2=(2.*DVTMN2-AUX(2)-PDVT)/(FLOAT(NPTS))
C
C
      PPDVT=AUX(2)
      PPHASE=0.0
      PPV=PPVPHA(CX(1),CX(2),ISNX)
      CX(1)=0.5*ALOG(AUX(1))
      CX(2)=0.0
      IO=1
      DO 50 I=2,IEND
      IO=IO+2
      IE=IO+1
      PDVT=AUX(IE)
      PPV=PPVPHA(CX(IO),CX(IE),ISNX)
      PHASE=PHAUNW(X,NX,ISNX,I,PPHASE,PPDVT,PPV,PDVT,ISSUC)
C
C  IF PHASE ESTIMATE SUCCESSFUL ,CONTINUE
C
      IF(ISSUC)GOTO 40
      ISSUC=.FALSE.
      RETURN
40    PPDVT=PDVT
      PPHASE=PHASE
      CX(IO)=0.5*ALOG(AUX(IO))
      CX(IE)=PHASE
D     TYPE *,IE,CX(IE)
      TYPE 41,'033,'133,'101,IFIX(FLOAT(I)/FLOAT(IEND)*100)
41     FORMAT('  ',3A1,I3,' %')
50    CONTINUE
C
C  REMOVE LINEAR PHASE COMPONENT
C
      ISFX=(ABS(PHASE/PI)+.1)
      IF (PHASE.LT.0.0)ISFX=-ISFX
      H=PHASE/FLOAT(NPTS)
      IE=0
      DO 60 I=1,IEND
      IE=IE+2
      CX(IE)=CX(IE)-H*FLOAT(I-1)
60    CONTINUE
C
C COMPUTE THE COMPLEX CEPSTRUM
C
      CALL FSST(CX,NFFT)
      RETURN
      END
C
C
      SUBROUTINE SPCVAL(NX,X,FREQ,XR,XI,YR,YI)
```

```
      DIMENSION X(1)
      DOUBLE PRECISION U0,U1,U2,W0,W1,W2,A,B,C,D,A1,A2,
     1SA0,CA0,XJ
C
C
C     INIT
C
      CA0=DBLE(COS(FREQ))
      SA0=DBLE(SIN(FREQ))
      A1=2.D+0*CA0
      U1=0.D+0
      U2=U1
      W1=U1
      W2=U1
      DO 10 J=1,NX
      XJ=DBLE(X(J))
      U0=XJ+A1*U1-U2
      W0=(DBLE(FLOAT(J-1)))*XJ+A1*W1-W2
      U2=U1
      U1=U0
      W2=W1
      W1=W0
10    CONTINUE
C
C
C
      A=U1-U2*CA0
      B=U2*SA0
      C=W1-W2*CA0
      D=W2*SA0
      A2=DBLE(FREQ*FLOAT(NX-1))
      U1=DCOS(A2)
      U2=-DSIN(A2)
      XR=SNGL(U1*A-U2*B)
      XI=SNGL(U2*A+U1*B)
      YR=SNGL(U1*C-U2*D)
      YI=SNGL(U2*C+U1*D)
      RETURN
      END
C
C
      FUNCTION PHAUNW(X,NX,ISNX,I,PPHASE,PPDVT,PPV,PDVT,ISCONS)
C
C     ROUTINE TO DO PHASE UNWRAPPING
C
      DIMENSION SDVT(17),SPPV(17),X(1)
      INTEGER SINDEX(17),PINDEX,SP
      LOGICAL ISCONS,FIRST
      COMMON PI,TWOPI,THLINC,THLCON,NFFT,NPTS,N,L,H,H1,DVTMN2
C
C
C
      FIRST=.TRUE.
      PINDEX=1
      SP=1
      SPPV(SP)=PPV
      SDVT(SP)=PDVT
C
C
```

```
      SINDEX(SP)=L+1
C
      GO TO 40
C
10    PINDEX=SINDEX(SP)
      PPHASE=PHASE
      PPDVT=SDVT(SP)
      SP=SP-1
      GO TO 40
C
C
20    IF((SINDEX(SP)-PINDEX).GT.1)GO TO 30
      ISCONS=.FALSE.
      PHAUNW=0.
      RETURN
C
CCC
C
30    K=(SINDEX(SP)+PINDEX)/2
C
      FREQ=TWOPI*(FLOAT(I-2)*FLOAT(L)+FLOAT(K-1))/H
      CALL SPCVAL(NX,X,FREQ,XR,XI,YR,YI)
C
C
      SP=SP+1
      SINDEX(SP)=K
      SPPV(SP)=PPVPHA(XR,XI,ISNX)
      XMAG=AMODSQ(XR,XI)
      SDVT(SP)=PHADVT(XR,XI,YR,YI,XMAG)
C
40    DELTA=H1*FLOAT(SINDEX(SP)-PINDEX)
      PHAINC=DELTA*(PPDVT+SDVT(SP))
CC
CC
C
      IF(ABS(PHAINC-DELTA*DVTMN2).GT.THLINC)GOTO 20
C
      PHASE=PPHASE+PHAINC
      CALL PHCHCK(PHASE,SPPV(SP),ISCONS)
      IF (.NOT.ISCONS)GOTO 20
C
CC
C
      IF(ABS(PHASE-PPHASE).GT.PI)GO TO 20
C
C
      IF(SP.NE.1)GOTO 10
      PHAUNW=PHASE
      RETURN
      END
C
C
C
      FUNCTION PPVPHA(XR,XI,ISNX)
      IF(XR.EQ.0.0 .AND. XI .EQ. 0.0) PPVPHA=0.0
      IF(XR.EQ.0.0 .AND. XI .EQ. 0.0) RETURN
            IF(ISNX.EQ.1) PPVPHA=(ATAN2(XI,XR))
            IF(ISNX.EQ.(-1)) PPVPHA=(ATAN2(-(XI),-(XR)))
      RETURN
      END
```

```
C
C
C
      FUNCTION PHADVT(XR,XI,YR,YI,XMAG)
      IF(XMAG .EQ. 0.0) PHADVT=0.0
      IF(XMAG .EQ. 0.0) RETURN
      PHADVT=-SNGL((DBLE(XR)*DBLE(YR)+DBLE(XI)*DBLE(YI))/DBLE(XMAG))
      RETURN
      END
C
C
C
      FUNCTION AMODSQ (ZR,ZI)
      AMODSQ=SNGL(DBLE(ZR)*DBLE(ZR)+DBLE(ZI)*DBLE(ZI))
      RETURN
      END
C
C
C
      SUBROUTINE PHCHCK(PH,PV,ISCONS)
C
C     THIS ROUTINE
C     CHECKS THE PHASE DIFFERENCE BETWEEN THE PREVIOUS DATA POINTS AND
C     IF THE PHASE DIFF DOES NOT EXCEED THE USER DEFINED THRESHOLD
C     THEN THE PHASE VALUE ISN'T CHANGED.
C
      COMMON PI,TWOPI,THLINC,THLCON,NFFT,NPTS,N,L,H,H1,DVTMN2
      LOGICAL ISCONS
C
      A0=(PH-PV)/TWOPI
      A1=FLOAT(IFIX(A0))*TWOPI+PV
      A2=A1+SIGN(TWOPI,A0)
      A3=ABS(A1-PH)
      A4=ABS(A2-PH)
C CHECK CONSISTENCY
      ISCONS=.FALSE.
      IF(A3.GT.THLCON.AND.A4.GT.THLCON)RETURN
      ISCONS=.TRUE.
C
      PH=A1
      IF(A3.GT.A4)PH=A2
      RETURN
      END
C
C
C
      SUBROUTINE ICEPS(CX,ISNX,ISFX)
      DIMENSION CX(2)
      COMMON PI,TWOPI,THLINC,THLCON,NFFT,NPTS,N,L,H,H1,DVTMN2
CCCC
CC
C
      SNX=FLOAT(ISNX)
      SFX=H/2.
      CX(NFFT+1)=0.
      CX(NFFT+2)=0.
      CALL FAST(CX,NFFT)
      CX(1)=SNX*EXP(CX(1))
C
C  PERFORM EXPONENTIATION OF TRANSFORMED DATA
```

```
C
      DO 10 K=3,NFFT+1,2
      K1=K+1
      PHDLY=SFX*FLOAT(K-1)
      T=SNX*EXP(CX(K))
      CX(K)=T*COS(CX(K1)+PHDLY)
      CX(K1)=T*SIN(CX(K1)+PHDLY)
10    CONTINUE
C
C     NOW PERFORM INVERSE FOURIER TRANSFORM
C
      CX(NFFT+2)=0.
      CALL FSST(CX,NFFT)
      RETURN
      END


      PROGRAM TESTGT
C
C     This program is used to provide convolved data for use in
C     homomorphic_processing.the user provides an ultrasonic pulse to
C     be used as reference and a synthetic reflector series.the output
C     of this program is a synthetic flaw signal.synthetic reflector
C     series is defined at integer sAmples of time.
C
C     COMMON PI,TWOPI,THLINC,THLCON,NFFT,NPTS,N,L,H,H1,DVTMN2
      DIMENSION CX(1026),AUX(1026)
      LOGICAL*1 FILNAM(15),TRUE,FALSE
      DATA TRUE/.TRUE./ FALSE/.FALSE./

      CALL JTITLE('TESTGT',6,'$',2.0)
C
C     GET PULSE DATA FILE NAME
C
20    TYPE 25
25    FORMAT(' Enter ultrasonic reference data filename: ',$)
      CALL GETDFN(FILNAM)
      IF (OPNFIL(3,FILNAM,'R').NE.TRUE) GOTO 20
      CALL GETDAT(3,NP,TFCTR,CX,TRUE)
C
C     READ THE REFLECTOR SERIES DATA
C
26    TYPE 30
30    FORMAT(' Enter synthetic reflector series data filename: ',$)
      CALL GETDFN(FILNAM)
      IF (OPNFIL(2,FILNAM,'R').NE.TRUE) GOTO 26
      CALL GETDAT(2,NP1,TFCTR1,AUX,.TRUE.)
C
      NP2 = NP
      IF (NP1.GT.NP2) NP2 = NP1
      NP2 = NP2+NP2
90    TYPE 35
35    FORMAT(' Enter size of DFT,length must be a power of two:',$)
      NFFT = iPROMT(NP2)
C
C     Check for correct NFFT
C
      IF(NFFT.GE.(NP+NP1)) GOTO 40
         TYPE *,' For linear convolution dft length must be
     1> or = sum of data points in both files!'
```

```
      GOTO 90
C
C     Append the necessary zeroes for fft computation
C
90    INITL=NP+1
      IEND=NFFT+2
      DO 120 I=INITL,IEND
120     CX(I)=0.0
      INITL=NP1+1
      DO 125 I=INITL,IEND
125     AUX(I)=0.0
C
C     COMPUTE FFT
C
      CALL FAST(CX,NFFT)
      CALL FAST(AUX,NFFT)
C
C     Compute linear convolution . Make sure that the dft length
C     is greater than 'NP+NP1-1' and also a multiple of two.
C
      TYPE *,'Computing linear convolution...'

      DO 130 I=1,NFFT+1,2                    ! This loop computes the
        T1=CX(I)*AUX(I)-CX(I+1)*AUX(I+1)     ! product of two DFT's:
        T2=CX(I+1)*AUX(I)+CX(I)*AUX(I+1)     ! CX and AUX
        CX(I)=T1
        CX(I+1)=T2
130   CONTINUE
C
C     Take inverse fourier transform and write the convolution
C     results out.
C
      CALL FSST(CX,NFFT)
C
C
C     Write the data for plotting purposes
C
45    TYPE 50
50    FORMAT(/,' Enter output convolved time data filename: ',$)
      CALL GETDFN(FILNAM)
      IF (OPNFIL(4,FILNAM,'W').NE.TRUE) GOTO 45
      CALL PUTDAT(4,NFFT,TFCTR,CX)
      CALL EXIT
      END
```

```
      PROGRAM ADDNOI
C
C         This program accepts a predefined filename (generally data
C         from a reflector wave and computes its statistics. Based
C         upon these statistics and a SNR value input by the user,
C         noise data is produced. The statistics of the noise data
C         are then computed including the actual SNR ratio. The noise
C         data is Gausian distributed using central limit theorem.
C
C                                          - Gregg Liming      3/4/85
C
C     Files:
C               Input:  Reference file            (REF###.DAT)
C               Output: Noise only file           (REF###.Nxx)
C                                                  where xx = SNR
C               Input:  Impulse file              (IMP###.DAT)
C               Output: Impulse + noise file      (IMP###.Nxx)
C                                                  where xx = SNR
C
C     Document!,document! What would Dr B. say!       alb
C
      REAL*4 SIG(512),RNOIS(512),S(512),IMPULS(512),SUMNOI(512)
      LOGICAL*1 FILNAM(15),OTFLNM(15),TEMP(6),KEYPR,EXT(3)
      INTEGER IX(2)
      DATA IX/0,0/
C
      CALL JTITLE('ADDNOI',6,'*',1,2)
10    TYPE 30
30    FORMAT(/,' Enter time data input filename (used to compute noise
      TYPE 35
35    FORMAT(' statistics): ',$)
      CALL GETDFN(FILNAM)
      IF(OPNFIL(3,FILNAM,'R').NE. .TRUE.)GOTO 10
45    CALL GETDAT(3,NP,TFACTR,SIG,.TRUE.)
      DO 90 I=1,NP
90    S(I)=1.
C
C        Comput statistics for input file data
C
      CALL TALLY(SIG,S,TOTAL,AVERS,SDSIG,VMIN,VMAX,NP,1,IER)
      IF(IER.EQ.0)GOTO 105
          TYPE 100,IER
100       FORMAT(/,' ERROR NO. ',I2,' IN COMPUTING STATISTICS.')
          GO TO 200
105   TYPE 110,(FILNAM(I),I=1,15)
110   FORMAT(//,' STATISTICS FOR INPUT FILE: ',15A1,/)
      TYPE 120,AVERS,SDSIG
120   FORMAT(5X,'Average = ',G12.5,', Standard deviation = ',G12.5)
      TYPE 130
130   FORMAT(//,' Enter SNR (relative to input file data) to compute
     1 noise data : ')
      TYPE 131
131   FORMAT(' (INTEGER < 100)             ',$)
      ACCEPT *,DBDWN
```

```
      DBDWN=ABS(DBDWN)
      SDNOIS=EXP(ALOG(10.0)*(ALOG10(SDSIG)-DBDWN/20.0))
C
C        Generate random number data with a gaussian distribution
C        with standard deviation, SDNOIS, and mean = 0.0 (assumes
C        input file has no DC offset
C
      TYPE *,' '
      TYPE *,'   Now computing random generator seed.'
      TYPE *,'   Depress any key to continue.'
11    CALL RANDU(IX(1),IX(2),RNOIS(1))
      CALL CHECK(IDUMMY,KEYPR)
      IF(KEYPR .NE. .TRUE.) GOTO 11
      DO 140 I=1,NP
      CALL GAUSS(IX,SDNOIS,0.0,RNOIS(I))
140   CONTINUE
C
C        Get statistics for noise data
C
      CALL TALLY(RNOIS,S,TOTAL,AVERN,SDNOIS,VMIN,VMAX,NP,1,IER)
      IF(IER.EQ.0)GOTO145
          TYPE 100,IER
          GO TO 200
145   SDR=20*ALOG10(SDNOIS/SDSIG)
      TYPE 150
150   FORMAT(//,' NOISE STATISTICS : ',/)
      SDR = -SDR
      TYPE 160,AVERN,SDNOIS,SDR
160   FORMAT(5X,'Average = ',G12.5,', Standard deviation = ',G12.5,/
     1,5X,'Actual SNR = ',G12.5,' dB')
      EXT(1) = 'N'
C      IDBDWN = INT(DBDWN) + 100 ! needed since ITOA pads w/ null char
      IDBDWN = INT(DBDWN)      !  Convert SNR to text
      CALL ITOA(IDBDWN,TEMP)   !  Get SNR in dB's into TEMP
      EXT(2) = TEMP(5)         !  Put SNR in dB's at last of filename
      EXT(3) = TEMP(6)         !  first 4 chars. are stripped off
      CALL FILEXT(FILNAM,EXT)
      TYPE 170,FILNAM
170   FORMAT(/,' Attempting to write noisy data to ',15A)
      IF(OPNFIL(2,FILNAM,'W') .EQ. .TRUE.) GOTO 260
280   TYPE 270
270   FORMAT(/,' Enter output filename : ',$)
      CALL GETDFN(FILNAM)
300   IF(OPNFIL(2,FILNAM,'W') .EQ. .FALSE.) GOTO 280
260   CALL PUTDAT(2,NP,TFACTR,RNOIS)
      TYPE *,'Data written.'
      TYPE *,' '
      TYPE 210
210   FORMAT(//,' Do you wish to add noise to an impulse (system) file
     1 ',$)
      IF (ASK('N') .EQ. .TRUE.) GOTO 200
225   TYPE 220
220   FORMAT(/,' Enter filename for impulse data : ',$)
      CALL GETDFN(FILNAM)
      IF (OPNFIL(3,FILNAM,'R') .EQ. .FALSE.) GOTO 225
      CALL GETDAT(3,NP2,TFACTR,IMPULS,.TRUE.)
      IF (NP2 .LE. NP) GOTO 227
        TYPE 228
228     FORMAT(/,' Number of data points is to large!! Try again.')
        GOTO 225
```

```
227     DO 250 I = 1,NP2
           SUMNOI(I) = IMPULS(I) + RNOIS(I)
250     CONTINUE
        CALL F(LEXT,FILNAM,EXT)
        TYPE 175,FILNAM
175     FORMAT(/,' Attempting to write impulse + noise data to ',13A,
223     IF (OPNFIL(4,FILNAM,'W') .EQ. .TRUE.) GOTO 221
        TYPE 222
222     FORMAT(1X,' Enter filename : ',$)
        CALL GETDFN(FILNAM)
        GOTO 223
221     CALL PUTDAT(4,NP2,TFACTR,SUMNOI)
        TYPE *,'Data stored.'
200     CONTINUE
        CALL EXIT
        END
C
C
C       ..................................................................
C
C          SUBROUTINE TALLY
C
C          PURPOSE
C             CALCULATE TOTAL, MEAN, STANDARD DEVIATION, MINIMUM, MAXIMUM
C             FOR EACH VARIABLE IN A SET (OR A SUBSET) OF OBSERVATIONS
C
C          USAGE
C             CALL TALLY(A,S,TOTAL,AVER,SD,VMIN,VMAX,NO,NV,IER)
C
C          DESCRIPTION OF PARAMETERS
C             A      - OBSERVATION MATRIX, NO BY NV
C             S      - INPUT VECTOR INDICATING SUBSET OF A. ONLY THOSE
C                      OBSERVATIONS WITH A NON-ZERO S(J) ARE CONSIDERED.
C                      VECTOR LENGTH IS NO.
C             TOTAL  - OUTPUT VECTOR OF TOTALS OF EACH VARIABLE. VECTOR
C                      LENGTH IS NV.
C             AVER   - OUTPUT VECTOR OF AVERAGES OF EACH VARIABLE. VECTOR
C                      LENGTH IS NV.
C             SD     - OUTPUT VECTOR OF STANDARD DEVIATIONS OF EACH
C                      VARIABLE. VECTOR LENGTH IS NV.
C             VMIN   - OUTPUT VECTOR OF MINIMA OF EACH VARIABLE. VECTOR
C                      LENGTH IS NV.
C             VMAX   - OUTPUT VECTOR OF MAXIMA OF EACH VARIABLE. VECTOR
C                      LENGTH IS NV.
C             NO     - NUMBER OF OBSERVATIONS
C             NV     - NUMBER OF VARIABLES FOR EACH OBSERVATION
C             IER    - ZERO, IF NO ERROR.
C                    - 1,IF S IS NULL. VMIN=1.E37,VMAX=-1.E37.,SD=AVER=0
C                    - 2, IF S HAS ONLY ONE NON-ZERO ELEMENT. VMIN=VMAX.
C                      SD=0.0
C
C          REMARKS
C             NONE
C
C          SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C             NONE
C
C          METHOD
C             ALL OBSERVATIONS CORRESPONDING TO A NON-ZERO ELEMENT IN S
C             VECTOR ARE ANALYZED FOR EACH VARIABLE IN MATRIX A.
```

```
C           TOTALS ARE ACCUMULATED AND MINIMUM AND MAXIMUM VALUES ARE
C           FOUND. FOLLOWING THIS, MEANS AND STANDARD DEVIATIONS ARE
C           CALCULATED.  THE DIVISOR FOR STANDARD DEVIATION IS ONE LESS
C           THAN THE NUMBER OF OBSERVATIONS USED.
C
C
C     ..................................................................
C
      SUBROUTINE TALLY(A,S,TOTAL,AVER,SD,VMIN,VMAX,NO,NV,IER)
      DIMENSION A(1),S(1),TOTAL(1),AVER(1),SD(1),VMIN(1),VMAX(1)
C
C        CLEAR OUTPUT VECTORS AND INITIALIZE VMIN,VMAX
C
      IER=0
      DO 1 K=1,NV
      TOTAL(K)=0.0
      AVER(K)=0.0
      SD(K)=0.0
      VMIN(K)=1.0E37
    1 VMAX(K)=-1.0E37
C
C        TEST SUBSET VECTOR
C
      SCNT=0.0
      DO 7 J=1,NO
      IJ=J-NO
      IF(S(J)) 2,7,2
    2 SCNT=SCNT+1.0
C
C        CALCULATE TOTAL, MINIMA, MAXIMA
C
      DO 6 I=1,NV
      IJ=IJ+NO
        X=A(IJ)
      TOTAL(I)=TOTAL(I)+X
      IF(X-VMIN(I)) 3,4,4
    3 VMIN(I)=X
    4 IF(X-VMAX(I)) 6,6,5
    5 VMAX(I)=X
    6 SD(I)=SD(I)+X*X
    7 CONTINUE
C
C        CALCULATE MEANS AND STANDARD DEVIATIONS
C
      IF (SCNT)8,8,9
    8 IER=1
      GO TO 15
    9 DO 10 I=1,NV
   10 AVER(I)=TOTAL(I)/SCNT
      IF (SCNT-1.0) 13,11,13
   11 IER=2
      DO 12 I=1,NV
   12 SD(I)=0.0
      GO TO 15
   13 DO 14 I=1,NV
   14 SD(I)=SQRT(ABS((SD(I)-TOTAL(I)*TOTAL(I)/SCNT)/(SCNT-1.0)))
   15 RETURN
      END
C
C     ..................................................................
C
```

```
C       SUBROUTINE GAUSS
C
C       PURPOSE
C          COMPUTES A NORMALLY DISTRIBUTED RANDOM NUMBER WITH A GIVEN
C          MEAN AND STANDARD DEVIATION
C
C       USAGE
C          CALL GAUSS(IX,S,AM,V)
C
C       DESCRIPTION OF PARAMETERS
C          IX -IX IS AN INTEGER ARRAY OF LENGTH 2.  THE INITIAL ENTRIES
C                IN THE IX ARRAY SHOULD BE ZERO.   THEREAFTER, IT WILL
C                CONTAIN PART OF A UNIFORMLY DISTRIBUTED INTEGER RANDOM
C                NUMBER GENERATED BY THE SUBROUTINE FOR USE ON THE
C                NEXT ENTRY TO THE SUBROUTINE.
C          S  -THE DESIRED STANDARD DEVIATION OF THE NORMAL
C                DISTRIBUTION.
C          AM -THE DESIRED MEAN OF THE NORMAL DISTRIBUTION
C          V  -THE VALUE OF THE COMPUTED NORMAL RANDOM VARIABLE
C
C       REMARKS
C          THIS SUBROUTINE USES RANDU WHICH IS MACHINE SPECIFIC
C
C       SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C          RANDU
C
C       METHOD
C          USES 12 UNIFORM RANDOM NUMBERS TO COMPUTE NORMAL RANDOM
C          NUMBERS BY CENTRAL LIMIT THEOREM. THE RESULT IS THEN
C          ADJUSTED TO MATCH THE GIVEN MEAN AND STANDARD DEVIATION.
C          THE UNIFORM RANDOM NUMBERS COMPUTED WITHIN THE SUBROUTINE
C          ARE FOUND BY THE POWER RESIDUE METHOD.
C
C     ..................................................................
C
      SUBROUTINE GAUSS(IX,S,AM,V)
      DIMENSION IX(2)
      A = 0.0
      DO 50 I=1,12
      CALL RANDU(IX(1),IX(2),Y)
   50 A=A+Y
      V=(A-6.0)*S+AM
      RETURN
      END
```

```
C     MODIFIED FOR RT-11 V4      22-MAR-84
C
      PROGRAM IMPULS
C*************************************************************************
C THIS PROGRAM ALLOWS ONE TO GENERATE AN IMPULSE TRAIN WITH ARBITRAY
C NUMBER OF SPIKES.THE CREATED IMPULSE DATA IS IN TIME DOMAIN.
C*************************************************************************
      DIMENSION DATA(1024)
      LOGICAL*1 FILNAM(18),IANS
C
 10   CONTINUE
 5    TYPE 16
 16   FORMAT(1X,' ENTER IMPULSE DATA FILENAME: ',$)
      CALL GETDFN(FILNAM)
      IF (OPNFIL(3,FILNAM,'W') .NE. .TRUE.) GOTO 5

      TYPE 30
30    FORMAT(/,' SELECT NUMBER OF DATA POINTS (2**I): ',$)
      ACCEPT *,NP
      RNP=NP
      WRITE(3)RNP
      TYPE 40
40    FORMAT(/,' HOW MANY SPIKES IN FULL RECORD? ',$)
      ACCEPT *,NS
      TYPE 50
50    FORMAT(/,' ENTER DELTA TIME FACTOR (DEFAULT=.80321285E-8): ',$)
      TMFCTR=.8032128514E-8
      ACCEPT *,TMFCTR
      WRITE(3)TMFCTR
      DO 60 I=1,NP
60    DATA(I)=0.0
      DO 100,I=1,NS
      TYPE 90
90    FORMAT(' ENTER SPIKE LOCATION (N) AND MAGNITUDE (REAL): ',$)
      ACCEPT *,N,DATA(N)
100   CONTINUE
      DO 110 I=1,NP
      WRITE(3)DATA(I)
110   CONTINUE
200   CLOSE(UNIT=3,DISPOSE='KEEP')
      TYPE 210
210   FORMAT(/,' WANT TO CREATE ANOTHER FILE? (Y OR N): ',$)
      ACCEPT 220,IANS
220   FORMAT(A1)
      IF(IANS .EQ. 'Y')GOTO 10
      END
```

```
      PROGRAM TRNSFM
C     THIS PROGRAM PROVIDES FOR DISCRETE FOURIER TRANSFORMATION OF
C     DATA IN A USER INTERACTIVE MODE.OUTPUT DATA IS AVAILABLE IN
C     IN AN UNFORMATTED FORM.THE DATA IS HEADED BY NUMBER OF
C     POINTS AND SEPARATION BETWEEEN FRQUENCY POINTS,FOLLOWED BY EXPON-
C     ENTIAL WEIGHT USED.THE TRANSFORMED DATA IS AVAILABLE WITH REAL
C     PART IN ODD RECORDS AND IMAGINARY PART IN EVEN NUMBERED RECORDS.
C
C     THIS DATA IS IN A FORM SUITABLE FOR PLOTTING ON THE HP PLOTTER
C     7225A USING 'FRQPLT' SUBROUTINE.
C
C     SUBROUTINES USED ARE 'FAST' WHICH IS A COMPLETE PACKAGE PROVIDING
C     FOR BOTH FORWARD AND INVERSE TRANSFORMATION OF DATA.NOTE HOWEVER
C     THAT IN ITS PRESENT FORM THE PROGRAM REQUIRES REAL(FUNCTION OF
C     SINGLE VARIABLE)DATA INPUT.THE FFT SIZE ALLOWED IS 1024 ALTHOUGH
C     THE FAST ROUTINE PROVIDES FOR UPTO 4096 POINTS.FAST IS AVAILABLE
C     AS A LIBRARY SUBROUTINE PACKAGE.
C
      DIMENSION X(1030),AUX(1030)
      LOGICAL*1 FILNAM(18), IANS
C
1     DO 5 I=1,1030
      X(I)=0.0
      AUX(I)=0.0
5     CONTINUE
10    TYPE 15
15    FORMAT(/,' FREQUENCY DATA OR TIME DATA?(F OR T)',$)
      ACCEPT 1100,IANS
1100  FORMAT(A1)
      IF(IANS .EQ.'F')GOTO 120
      IF(IANS .NE.'T') GO TO 16
      IF(IANS .EQ.'T') GO TO 17
16          TYPE *,' WRONG DATA TYPE,TRY AGAIN? '
            GO TO 10
17    CONTINUE
C
C     OPEN TIME DATA FILE
C
18    TYPE 20
20    FORMAT(/,' ENTER INPUT TIME DATA FILE NAME: ',$)
      CALL GETDFN(FILNAM)
      IF (OPNFIL(3,FILNAM,'R') .NE. .TRUE.) GOTO 18
      READ(3)RNP
      NP=RNP
      TYPE 1200,NP
1200  FORMAT(/,' NUMBER OF DATA POINTS = ',I4,/)
      READ(3)XFCTR
      TYPE 1300,XFCTR
1300  FORMAT(/,' DATA SPACING IS :',G12.6)
      DO 50 I=1,NP
      READ(3,ERR=55)X(I)
50    CONTINUE
      GOTO 60
55    TYPE *,' READ ERROR, FILE SCREWED UP'
```

```
        GO TO 200
60      CLOSE(UNIT=3,DISPOSE='KEEP')
C
C       PROMPT THE USER FOR EXPONENTIAL WEIGHTING FACTOR WHICH SHOULD BE
C       AS CLOSE AS POSSIBLE TO 1.0 (0.9999 IS A GOOD CHOICE FOR NOISE
C       FREE DATA).
C
        EW=1.0
        TYPE 65
65      FORMAT(/,' ENTER WEIGHTING FACTOR TO BE USED ( <1.0 ):',$)
        ACCEPT *,EW
        CALL EXPWAT(X,EW,NP,1)
C
C       DEFINE FFT SIZE AND CHECK FOR CORRECT NFFT TO AVOID ALIASING.
C
70      TYPE 72
72      FORMAT(/,' ENTER DFT SIZE, LENGTH MUST BE A POWER OF TWO :'$)
        ACCEPT *,NFFT
C
        IF(NFFT.LT.NP) GO TO 73
        IF(NFFT.GE.NP) GO TO 74
73              TYPE *,' DFT LENGTH MUST BE > NUMBER OF DATA POINTS:'
                GOTO 70
74      CONTINUE
C

        FMAX=1./(2.*XFCTR)
        FFCTR=2.*FMAX/FLOAT(NFFT)
C
C       GET READY TO COMPUTE FFT USING 'FAST'
C

C       FORM SEQUENCES FROM X(N) AND N*X(N) FOR DFT.
C
        DO 90 I=1,NP
        AUX(I)=FLOAT(I-1)*X(I)
90      CONTINUE
C
C       COMPUTE FFT
C
        CALL FAST(X,NFFT)
        CALL FAST(AUX,NFFT)
C
C       STORE THIS FREQUENCY DOMAIN DATA.
C
C       OPEN A NEW FILE WITH GIVEN NAME.THE FIRST RECORD CONTAINS NUMBER OF
C       FREQUENCY POINTS AND FREQUENCY SPACING.SUBSEQUENT RECORDS CONTAIN
C       REAL AND IMAGINARY PART OF TRANSFORMED DATA.FOLLOWED BY REAL
C       &IMAGINARY PART OF N TIMES TRANSFORMED DATA.
C
104     TYPE 105
105     FORMAT(/,' ENTER OUTPUT FILE NAME(FREQUENCY DATA)',$)
        CALL GETDFN(FILNAM)
        IF (OPNFIL(2,FILNAM,'W') .NE. .TRUE.) GOTO 104
        REAL=1.+FLOAT(NFFT)/2.
        WRITE(2)REAL,FFCTR,EW,FMAX
        DO 110 I=1,NFFT+1,2
        WRITE(2)X(I),X(I+1),AUX(I),AUX(I+1)
110     CONTINUE
        CLOSE(UNIT=2,DISPOSE='KEEP')
```

```
      GO TO 200
C
C     PROCESSING FREQUENCY DOMAIN DATA
C
120   TYPE 125
125   FORMAT(/,' ENTER FREQUENCY DATA FILE NAME ( INPUT DATA):',$)
      CALL GETDFN(FILNAM)
      IF (OPNFIL(3,FILNAM,'R') .NE. .TRUE.) GOTO 120
      READ(3)RNP,FFCTR,EW,FMAX
      NP=RNP
      IF(EW.EQ.0.0)EW=1.0
      TYPE 1200,NP
      TYPE 1300,FFCTR
      TYPE 1400,EW
1400  FORMAT(/,' EXPONENTIAL WEIGHT USED IS :',F6.4)
      DO 140 I=1,NP
      J=2*I-1
      READ(3,ERR=55)X(J),X(J+1),AUX(J),AUX(J+1)
140   CONTINUE
      CLOSE(UNIT=3,DISPOSE='KEEP')
150   TYPE 72
      ACCEPT *,NFFT
      IF(NFFT.LT.2*(NP-1)) GO TO 151
      IF(NFFT.GE.2*(NP-1)) GO TO 152
151      TYPE *,' DFT SIZE MUST BE >2*(NUMBER OF FREQ POINTS-1)'
         GOTO 150
152   CONTINUE
      TMAX=1./FFCTR
C
C     PERFORM INVERSE TRANSFORMATION TO RECOVER REAL DATA FROM
C     FFT TRANSFORMED DATA.
C
170   CALL FSST(X,NFFT)
      CALL EXPWAT(X,EW,NFFT,1)
C
C     STORE THIS DATA
C     OPEN A NEW FILE WITH GIVEN NAME.THE FIRST RECORD CONTAINS NUMBER O
C     DATA POINTS AND SECOND CONTAINS SAMPLING INTERVAL FOLLOWED BY THE
C     DATA.USE 'HPPLOT' TO PLOT THIS DATA.
C
174   TYPE 175
175   FORMAT(/,' ENTER OUTPUT TIME DATA FILE NAME :',$)
      CALL GETDFN(FILNAM)
      IF (OPNFIL(3,FILNAM,'W') .NE. .TRUE.) GOTO 174
      REAL=FLOAT(NFFT)
      XFCTR=TMAX/FLOAT(NFFT)
      WRITE(2)REAL
      WRITE(2)XFCTR
      DO 180 I=1,NFFT
      WRITE(2)X(I)
180   CONTINUE
      CLOSE(UNIT=2,DISPOSE='KEEP')
200   TYPE 210
210   FORMAT(/,' DO YOU WANT TO TRY AGAIN?(Y OR N): '$)
      IANS='Y'
      ACCEPT 1100,IANS
      IF(IANS .EQ. 'Y')GOTO 1
      END
C
      SUBROUTINE EXPWAT(X,A,N,K)
```

```
      DIMENSION X(2)
      IF (A.EQ.1.0)RETURN
      FX=1.
      DO 10 I=1,N
      X(I)=X(I)*FX
      IF(K .GT.0) GO TO 15
      IF(K.LE.0) GO TO 20
15            FX=FX*A
20            FX=FX/A
10    CONTINUE
      RETURN
      END
```

```
      PROGRAM WIENER
C
C     MODIFICATION HISTORY:
C
C     2?-Feb-85          Check NFFT against largest of (np1, np2)
C     NEEDS =            add time domain plot after transform
C     2?-Mar-85          Changed starting pt. in loops from 2 to 3
C                        and changed X() to REF(), Y() to FLAW(),
C                        I to REAL, & IMAG to make program flow easier
C                        to follow.
C
C     12-MAY-85          Add if one wants to plot the time domain result
C                        to the plotter.
C                                                                 AY.
C
C     16-Jul-85          Corrected YPLOMF & General update        alb
C
C
C     THIS PROGRAM PERFORMS WIENER FILTERING ON THE DATA IN FREQUENCY
C     DOMAIN ACCORDING TO THE EQUATION R'(f)*F(f)/(R(F)**2+FACTR).
C     FACTR IS THE DESENSITIZING COEFFICIENT AND IS USUALLY SET TO
C     REDUCE THE ABOVE DIVISION TO A SMALL VALUE OUTSIDE THE RANGE OF
C     INTEREST,I.E.,OUTSIDE THE FREQUENCY BAND OF THE TRANSDUCER.
C
      LOGICAL*1 FILNAM(15),REFFIL(15),FLAWFL(15),ANSWER
      REAL*4 FFCTR,XFCTR,XFCTR2,RMAX,CUTOFF,PERCNT,TMFCTR
      INTEGER*2 REAL,IMAG,I,NP1,NP2,NP3
      DIMENSION REF(1030),FLAW(1030)
C
    CALL JTITLE('WIENER',6,'$',2,1)
C
10    DO 20 I=1,1030
        REF(I)=0.0
        FLAW(I)=0.0
20    CONTINUE
C
C     GET DATA FROM FILES
C
      CALL GETD(REFFIL,FLAWFL,NP1,NP2,XFCTR,XFCTR2,REF,FLAW)
      TMFCTR=XFCTR
C
C     PROMPT THE USER FOR EXPONENTIAL WEIGHTING FACTOR WHICH SHOULD BE
C     AS CLOSE AS POSSIBLE TO 1.0 (0.9999 IS A GOOD CHOICE FOR NOISE
C     FREE DATA).
C
      EW=1.0
      TYPE 30
30    FORMAT(/,' Enter weighting factor to be used ( <1.0 ): ',$)
      ACCEPT *,EW
      CALL EXPWAT(REF,EW,NP1,1)
      CALL EXPWAT(FLAW,EW,NP2,1)
C
C     DEFINE FFT SIZE AND CHECK FOR CORRECT NFFT TO AVOID ALIASING.
C
```

```
         NPD = NP2
         IF (NP1.GT.NP2) NPD = NP1          ! Compare to largest
40       TYPE 50
50       FORMAT(/,' Enter DFT size, length must be a power of two? ',$)
         NFFT = IPROMT(NPD)
C
         IF(NFFT.GE.NPD) GO TO 60
                TYPE *,' DFT length must be > number of data points'
                GOTO 40
60       CONTINUE
C

         FMAX=1./(2.*XFCTR)
         FFCTR=2.*FMAX/FLOAT(NFFT)
C
C        COMPUTE FFT
C
         TYPE *,' '
         TYPE *,'Transforming Reference data to S-Plane...'
         CALL FAST(REF,NFFT)
         TYPE *,'Transforming Flaw data to S-Plane...'
         TYPE *,' '
         CALL FAST(FLAW,NFFT)
C
         TYPE 70
70       FORMAT(/,' Do you wish to plot the freq. domain data ',$)
         IF (ASK('N') .EQ. .TRUE.) GOTO 120
80       TYPE 90
90       FORMAT(/,' Do you wish to plot the Ref or Flawed data (R/F)? ',$)
95       ACCEPT 100,ANSWER
100      FORMAT(1A)
         IF (.NOT.(ANSWER.EQ.'F'.OR.ANSWER.EQ.'R')) GOTO 95
         IF (ANSWER.EQ.'F') CALL YPLOMF(FLAW,FFCTR,NFFT,FLAWFL)
         IF (ANSWER.EQ.'R') CALL YPLOMF(REF,FFCTR,NFFT,REFFIL)
         TYPE 110
110      FORMAT(/,' Another plot ',$)
         IF (ASK('N') .NE. .TRUE.) GOTO 80
120      CONTINUE
C
C        WIENER TRANSFORM
C
         FLAW(1)=0.0                        ! set d.c. component to zero
         FLAW(2)=0.0
         REF(1)=0.0
         REF(2)=0.0
         DO 130 REAL=3,NFFT,2
           IMAG = REAL + 1
           TEMP1=REF(REAL)*FLAW(REAL)+REF(IMAG)*FLAW(IMAG)
           TEMP2=REF(REAL)*FLAW(IMAG)-REF(IMAG)*FLAW(REAL)
           FLAW(REAL)=TEMP1
           FLAW(IMAG)=TEMP2
130      CONTINUE
C
C        COMPUTE SQUARED MAGNITUDE OF REFERENCE DATA
C
         RMAX=0.0
         DO 140 REAL=3,NFFT,2
           IMAG = REAL + 1
           REF(REAL)=REF(REAL)**2+REF(IMAG)**2
           IF(RMAX.LT.REF(REAL))RMAX=REF(REAL)
```

```
140     CONTINUE
C
C       GET FROM USER AMPLITUDE CUTOFF POINT
C
        TYPE 150
150     FORMAT(/,'Enter cutoff amplitude percentage to supress noise
     1 ratios:',$)
        PERCNT=iPROMT(10)
        PERCNT=PERCNT/100.
        CUTOFF=PERCNT*SQRT(RMAX)
C
C       IF MAGNITUDE IS LESS THAN CUTOFF SET THE RATIO TO 0.0
C
        DO 170 REAL=3,NFFT,2
          IMAG = REAL + 1
          IF(SQRT(REF(REAL)).GE.CUTOFF) GOTO 160
                FLAW(REAL)=0.0                  ! If below cutoff, erase
                FLAW(IMAG)=0.0
                GOTO 170
160       FLAW(REAL)=FLAW(REAL)/REF(REAL)
          FLAW(IMAG)=FLAW(IMAG)/REF(REAL)
170     CONTINUE
C
C       PLOT WIENER TRANSFORM DATA
C
        TYPE 180
180     FORMAT(/,' Wish to plot the flaw transfer function ',$)
        IF (ASK('Y') .EQ. .TRUE.) CALL YPLOMF(FLAW,FFCTR,NFFT,FLAWFL)
C
        TYPE 190
190     FORMAT(/,' Wish to save this transfer function ',$)
        IF (ASK('Y') .NE. .TRUE.) GOTO 260
C
C         TAKE INVERSE FOURIER TRANSFORM
C
        TYPE *,' '
        TYPE *,'Converting data to the time domain...'
        TYPE *,' '
        CALL FSST(FLAW,NFFT)
        CALL EXPWAT(FLAW,EW,NFFT,0)
C
C

200     TYPE 210
210     FORMAT(/,' Enter deconvolved data filename: ',$)
        CALL GETDFN(FILNAM)
        IF (OPNFIL(2,FILNAM,'W') .NE. .TRUE.) GOTO 200
        CALL PUTDAT(2,NFFT,XFCTR,FLAW)

        CALL CLEAR
        TYPE 220
220     FORMAT(/,' Do you wish to plot the data in the time domain ',$)
        IF (ASK('N') .NE. .TRUE.)CALL YPLOY(FLAW,NFFT,3,TMFCTR)
        TYPE 230
230     FORMAT(/,' Do you wish to transfer the graph to the plotter',$)
        IF (ASK('Y') .NE. .TRUE.) GO TO 260
        TYPE *,' '
        TYPE *,' Please set the plotter on line(release local button)'
        TYPE *,' Also set the printer to local mode.'
        TYPE *,' '
```

```
240     TYPE 250
250     FORMAT(' Ready ',$)
        IF (ASK('Y').EQ..FALSE.) GO TO 240
        CALL YPLOT(FLAW,NFFT,1,TMFCTR)
260     TYPE 270
270     FORMAT(/,' Do you want to try again '$)
        IF (ASK('Y') .EQ. .TRUE.) GOTO 10
        CALL EXIT
999     END
```

```
        PROGRAM DECONV
C
C       Wenner-Gren Lab.          Version 1.0                   slb
C
        LOGICAL*1 NCHAR,MCHAR
        DIMENSION X(515),Y(515),YFIT(515)
        REAL A(50,50),B(50),Z(200),C(50),R(51),P(500),D(50)
        LOGICAL*1 IANS
        INTEGER*2 TEKCRT,CRTAGN
        DATA TEKCRT/3/ CRTAGN/4/

        CALL JTITLE('TEKDEC',6,'=',1.0)
 5      CALL CLEAR
C
C       GET INPUT DATA FILE
C
        DO 7 I=1,256
        YFIT(I)=0.0
        Y(I)=0.0
7       CONTINUE
16      CALL INPUT(Y,NP,TFCTR,1)
        IF (NP .LE. 256) GOTO 15
          TYPE *,' Number of data points must be <= 256'
          GOTO 16
C
C       CREATE TIME AXIS
C
15      DO 20 I=1,NP
        X(I)=FLOAT(I-1)
20      CONTINUE
C
C       PROCEED TO GET PARAMETERS FOR SPLINE FITTING.
C
30      TYPE 40
40      FORMAT(/,' Enter ratio of knot-spacing to data spacing: ',$)
        ACCEPT *,KR
        DO 50 I=1,200
           Z(I)=FLOAT(I-1)*FLOAT(KR)
50         CONTINUE
        TYPE 60
60      FORMAT(/,' Enter order of splines: ',$)
        ACCEPT *,KK
        TYPE 70
70      FORMAT(/,' Enter number of basic splines: ',$)
        ACCEPT *,N
        KK1=KK-1
           DO 90 I=0,KK1
           S=0.
           LU=KR*(KK-I)
              DO 80 L=0,LU
              T1=FLOAT(L+KR*I)
              T2=FLOAT(L)
              S= S+BSP(T1,Z,1,KK)*BSP(T2,Z,1,KK)
80            CONTINUE
           R(I+1)=S                                            ! R(I)
90         CONTINUE
```

```
      DO 120 I=1,KR
      LU=N-I+1
         DO 110 J=1,LU
         A(J,J+I-1)=R(I)                                      ! R(I-1)
110      CONTINUE
120   CONTINUE
         DO 140 J=1,N
         B(J)=0.
         ML=KR*(J-1)+1
         MU=KR*(J-1+KK)+1
            DO 130 I=ML,MU
            JKL1 = J
            B(J)=B(J)+Y(I)*BSP(X(I),Z,JKL1,KK)
130         CONTINUE
140      CONTINUE
      KB=KK-1
      CALL BWS(A,N,KB,B,C)
         DO 150 I=1,NP
         T=FLOAT(I-1)
         YFIT(I)=PS(T,Z,C,N,KK)
150      CONTINUE
         DO 155 I=1,NP
         X(I)=FLOAT(I-1)*TFCTR
155      CONTINUE
      TYPE 160
160   FORMAT(/,' Wish to plot fit to reference data? ',$)
      IF (ASK('Y') .NE. .TRUE.) GOTO 11
170   CALL CLEAR
      TYPE *,' '
      TYPE *,'Plotting reference data...'
      CALL PLOTXY(X,Y,NP,TEKCRT)
      TYPE *,' Plotting spline fitted data...'
      CALL PLOTXY(X,YFIT,NP,CRTAGN)
      TYPE 180
180   FORMAT(/,' Do you wish to replot this graph? ',$)
      IF (ASK('N') .EQ. .FALSE.) GOTO 170
C
C     BEGIN DECONVOLUTION STEPS.
C
11    DO 190 I=1,NP1
190   Y(I)=0.
C
C     GET SCATTERED DATA FILE NAME
C
195   CALL INPUT(Y,NP1,TFCTR,2)
      IF (NP .LE. 256) GOTO 196
        TYPE *,' Number of data points must be <= 256'
        GOTO 195
C
C     GET PARAMETERS FOR SCATTERED DATA FITTING.
C
196   TYPE 210
210   FORMAT(/,' Enter order of solution splines: ',$)
      ACCEPT *,KS
      TYPE 220
220   FORMAT(/,' Enter number of solution splines: ',$)
      ACCEPT *,L
      KN=KK+KS
      LL=(N+KN-1)*KR-1
      IF(LL.GT.500)TYPE*,'(N+KN-1)*KR TOO BIG...'
```

```
            DO 240 I=1,LL
            S=0.
            T=FLOAT(I)
               DO 230 L1=1,N
                  JLK2 = L1
               S=S+C(L1)*BSP(T,Z,JLK2,KN)
230            CONTINUE
            P(I)=S
240         CONTINUE
         LL=N+KN-2
         IF(KR*(N+L+KN-1).GE.M)TYPE*,'(N+L+KN-1)*KR TOO BIG...'
            DO 260 I=0,LL
            S=0.
            LI=(N+KN-I-1)*KR-1
               DO 250 LS=1,LI
               S=S+P(LS)*P(LS+I*KR)
250            CONTINUE
            R(I+1)=S                                              ! R(I)
260         CONTINUE
         DO 290 I=1,L
         DO 290 J=I,L
290      A(I,J)=0.0
         IF(LL.GE.L-1)LL=L-1
         DO 310 I=0,LL
         LI=L-I
            DO 300 J=1,LI
            A(J,J+I)=R(I+1)                                       ! R(I)
            JI=J+I
300         CONTINUE
310      CONTINUE
         KL=(N+KN-1)*KR-1
         DO 350 K1=1,L
         S=0.
         KL1=KL
         KL2=KL+(K1-1)*KR+1
         IF(KL2.GE.NP1)KL1=NP1-(K1-1)*KR-1
            DO 340 I=1,KL1
            S=S+Y(I+(K1-1)*KR+1)*P(I)
340         CONTINUE
         B(K1)=S
350      CONTINUE
         IBW=N+KN-2
         IF(IBW.GE.L-1)IBW=L-1
         CALL BWS(A,L,IBW,B,D)
            DO 360 I=1,NP1
            T=FLOAT(I-1)
360         YFIT(I)=FS(T,Z,D,L,KS)
         TYPE 365
365      FORMAT(/,' Plot computed impulse response? ',$)
         IF (ASK('Y') .NE. .TRUE.) GOTO 12
370      CALL CLEAR
         CALL PLOTXY(X,YFIT,NP1,TEKCRT)
         TYPE 180
         IF (ASK('N') .NE. .TRUE.) GOTO 370
12       TYPE 375
375      FORMAT(/,' Wish to store impulse response? '$)
         IF (ASK('Y') .NE. .TRUE.) GOTO 390
         TYPE 380
380      FORMAT(/,' Enter impulse response data file name ',$)
         CALL OUTPUT(YFIT,NP1,TFCTR)
```

```
C
C       START COMPUTATIONS OF FIT TO FLAW DATA.
C
390     DO 410 I=1,NP1
        LL=(N+KN-1)*KR - 1
        S=0.0
           DO 400 J=1,L
           IF(I-1-(J-1)*KR.LT.1)GO TO 400
           IF(I-1-(J-1)*KR.GT.LL)GO TO 400
           S=S+D(J)*P(I-1-(J-1)*KR)
400        CONTINUE
410     YFIT(I)=S
        TYPE 420
420     FORMAT(/,' Plot fit to flaw data ? ',$)
        IF (ASK('Y') .NE. .TRUE.) GOTO 590
570        CALL CLEAR
        CALL PLOTXY(X,Y,NP1,TEKCRT)
        CALL PLOTXY(X,YFIT,NP1,CRTAGN)
            TYPE 180
        IF (ASK('N') .NE. .TRUE.) GOTO 570
590     TYPE 600
600     FORMAT(/,' Wish to perform another miracle? ',$)
        IF (ASK('N') .NE. .TRUE.) GOTO 5
        CALL EXIT
999     STOP
        END
```

```
      SUBROUTINE GNOISE(U1,U2,ISEED)
C
C     Wright-Patterson Air Force Base
C
C     The subroutines used in this program were taken from:
C
C     'Programs for Digital Signal Processing'
C     IEEE Press, 1979
C     345 East 47 Strett, New York, NY 10017
C     Sponsored by the IEEE Acoustics, Speech, and
C                       Signal Processing Society
C     Lib. of Congress Cat. Card # 79-89028
C     IEEE Book # 0-87942-128-2 (paperback ver.)
C                # 0-87942-127-4 (hardback)
C     Also published by John Wiley & Sons, Inc.
C     Wiley Order # 0-471-05961-7 (paperback ver.)
C                  # 0-471-05962-5 (hardback)
C
      DIMENSION U1(1024),U2(1024)
      TWOPI=8.0*ATAN(1.0)
C
C     CALCULATE A UNIFORM DISTRIBUTION. SEED FOR U2 IS U1(1024)
C
C
      CALL UNIDST(U1,1024,ISEED)
      ISEED=U1(1024)*16384
      CALL UNIDST(U2,1024,ISEED)
C
C     NOW COMPUTE A NORMAL DISTRIBUTION FROM U1 AND U2.
C     PLACE RESULT IN U1.
C
      DO 10 I=1,1024
      U1(I)=SQRT(-2.0*ALOG(U1(I)))*COS(TWOPI*U2(I))
10    CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE UNIDST(U,N,ISEED)
      DIMENSION U(2)
CC
C
      IF (ISEED.EQ.0)GOTO 20
      IS=ISEED
      DO 10 I=1,N
      ISIS=MOD(131*IS,16384)
      U(I)=FLOAT(IS)/16384.0
10    CONTINUE
      RETURN
```

```
20      WRITE(4,30)
30      FORMAT(//' ERROR , SEED CANNOT BE ZERO ')
        STOP
        END
```

```
      PROGRAM CEPSTR
C
C     THIS PROGRAM READS AN INPUT DATA FILE FROM A DISC PREVIOUSLY DEFINE
C     THE DATA CONSISTS OF CONVOLVED SIGNAL IN TIME DOMAIN.EXPONENTIAL
C     WEIGHTING IS USED TO ELIMINATE SPECTRAL ZEROES.COMPLEX CEPSTRUM IS
C     COMPUTED USING A SET OF SUBROUTINES ADAPTED FROM LITERATURE.
C     PHASE UNWRAPPING IS ACCOMPLISHED THROUGH THE TECHNIQUE DEVISED BY
C     TRIBOLET.INTERACTIVE PLOTTING IS POSSIBLE ON H-P PLOTERS THROUGH US
C     OF THE 'HPISPP' PLOTTING PACKAGE.
C
C     Version 1.1              10-Jan-85
C     Version 1.2              24-Jul-85        Modified PLOTY
C
C
C     EDIT HISTORY:
C          Copied from HPCEPS and modified to its present form Feb. 13,19
C                                                                - GWL
C
      COMMON /DATA/Y(515),CY(515)
      COMMON PI,TWOPI,THLINC,THLCON,NFFT,NPTS,N,L,H,H1,DVTMN2
      DIMENSION AUX(515)
      LOGICAL*1 IANS,NPAGE(2),IPZER(10),ROTATE
      LOGICAL*1 FILNAM(15)
      INTEGER ITPRL(6),GW
      LOGICAL ISSUC
C
      CALL JTITLE('CEPSTR',6,'*',1.2)
C     INITIALIZE BUFFERS AND CONSTANTS
C

      ROTATE = .FALSE.
      THLINC=1.5       ! set compare values for phaunw
      THLCON=.5        !
      PI=4.0*ATAN(1.0)
      TWOPI=2.*PI
C
C     GET INPUT TIME DOMAIN DATA.
C
      TYPE 5
5     FORMAT(/,' Enter input time-domain data filename (convolved
     1 data) : ',$)
      CALL GETDFN(FILNAM)
      IF (OPNFIL(3,FILNAM,'R') .NE. .TRUE.) GOTO 7
      CALL GETDAT(3,NP,TFCTR,Y,.TRUE.)
C
      FFCTR=1./(TFCTR*FLOAT(NP))
C
C     GET THE SIZE OF DFT AND CHECK ITS LENGTH FOR PROPER COMPUTATIONS.
```

```
C
30      TYPE 32
32      FORMAT(/,' Enter size of DFT, Length must be a power of two ')
        TYPE 33
33      FORMAT(' and less than or equal to 512. ',$)
        NFFT = IFROMT(512)
C         Check for NFFT too large.
        IF(NFFT .GT. 512) GOTO 30
C         Check for NFFT not a power of two
        LOG2NP=0
        ITEMP=NFFT
34      IF(ITEMP .LE. 1) GOTO 508
            ITEMP=ITEMP/2
            LOG2NP=LOG2NP+1
            GO TO 34
508     NPPOW2=2**LOG2NP
        IF(NFFT .NE. NPPOW2) GOTO 30
C         NFFT must be a power of two
        RNFFT=FLOAT(NFFT)
        DO 35 I=1,NFFT÷2           !USE AUX FOR PLOT PURPOSES
          AUX(I)=(I-1)*TFCTR       ! AUX is used as X data
35      CONTINUE
C
C      Normalize data upon request
C
        TYPE 600
600     FORMAT(/,' Do you wish to normalize the data ',$)
        IF(ASK('N') .EQ. .TRUE.) GOTO 2
        CALL NORMAL(Y,NP,1)
C
C       PROMPT THE USER FOR EXPONENTIAL WEIGHTING FACTOR.
C       WEIGHTING FACTOR SHOULD BE AS CLOSE AS POSSIBLE TO 1.0.
C
2       EW=0.9999
31      TYPE 36
36      FORMAT(/,' Enter weighting factor ( Default = .9999 ) ')
        TYPE 640
640     FORMAT(1X,'Cepstral processing time is decreased with decreased
     1 weighting factor : ',$)
        ACCEPT *,EW
        IF (EW .EQ. 1.0) GOTO 512
        IF (EW .LT. 1.0) GOTO 39
        TYPE 45
45      FORMAT(/,' Exponential weighting is > 1. Are you sure ',$)
        IF(ASK('Y') .EQ. .FALSE.) GOTO 31
39          FX=1.
            DO 37 I=1,NP
              Y(I)=Y(I)*FX
              FX=FX*EW
37          CONTINUE
C
C       PLOT WEIGHTED DATA
C
512     TYPE 38
38      FORMAT(/,' Do you wish to plot the weighted data ',$)
        IF (ASK('N') .EQ. .TRUE.) GOTO 53
            CALL CLEAR
            CALL PLOTXY(AUX,Y,NP,3)          ! RT-11 PLOT
53      TYPE 111
111     FORMAT(/,' Do you wish to save weighted data ',$)
```

```
      IF (WARNING .EQ. .TRUE.) GOTO 31
      TYPE 34
34    FORMAT(/,' Enter weighted data filename : ',$)
      CALL GETDFN(FILNAM)
      IF (OPNFIL(3,FILNAM,'U') .NE. .TRUE.) GOTO 33
      CALL PUTDAT(3,NP,TFCTR,Y)
C
C     COMPUTE COMPLEX CEPSTRUM
C
314   TYPE *,' '
      TYPE *,'  ... Now computing complex cepstra. '
      TYPE *,' '
      T1 = SECNDS(0.)
      CALL CCEPS(NP,Y,ISNX,ISFX,ISSUC,CY,AUX)
      SCONDS = SECNDS(T1)
      IHOURS = INT(SCONDS/3600.0)
      MINITS = INT(SCONDS/60.0) - 60.0*IHOURS
      SCONDS = SCONDS - 60.0*FLOAT(MINITS) - 3600.0*FLOAT(IHOURS)
      TYPE 650,EW,IHOURS,MINITS,SCONDS
650   FORMAT(/,' Cepstral processing of the weighted data with a weigh
     1ting of ',F7.5,/,' required ',I2,' hrs., ',I2,' mins., ',F5.2,' se
     1cs. to process.')
C
      IF(ISSUC)GO TO 50
      TYPE 46
46    FORMAT(/,' Phase unwrapping failed. Try using a lower exponentia
     1l weighting. ')
      GO TO 2
C
C     PHASE UNWRAPPING SUCCESSFUL.  WRITE OUT POLARITY ,PHASE DELAY
C     AND EXPONENTIAL WEIGHTING USED.
C
50    TYPE 51,ISNX,ISFX,EW
51    FORMAT(/,' Sign = ',I2,/,' Linear phase = ',
     1I4,/,' Exponential weighting used = ',E10.3,/)
C
C       Compute total energy in cepstrum
C
      TOTENG=0.
      DO 60 I=1,NFFT
        TOTENG=TOTENG+CY(I)*CY(I)
        AUX(I)=CY(I)            ! save CY data for repeated use
60    CONTINUE
C
      TYPE 61,TOTENG
61    FORMAT(/,' Total energy in cepstrum = ',E12.3,' joules.')
C
C       Rotate data
C
83          DO 80 I=1,NFFT/2
              J=I+NFFT/2
              TEMP=CY(I)
              CY(I)=CY(J)
              CY(J)=TEMP
              Y(J)=(I-1)*TFCTR
              Y(I)=(I-(NFFT/2+1))*TFCTR
80          CONTINUE
      TYPE *,' '
      TYPE *,' ... Prepare to plot 'ungated' cepstral data. '
      TYPE *,' '
```

```
          CY(NFFT+1) = CY(NFFT)
          CALL CLEAR
          CALL PLOTY(CY,NFFT,3)          ! RT-11 PLOT
          TYPE 701
701       FORMAT(/,' Do you want to plot data on the plotter ',$)
          IF(ASK('N') .EQ. .TRUE.) GOTO 401
          CALL PLOTY(CY,NFFT,1)
C
401     TYPE 400
400     FORMAT(/,' Do you wish to save cepstra data ',$)
        IF (ASK('N') .EQ. .TRUE.) GOTO 415
C  Put cepstra data into FILNAM
402       TYPE 405
405       FORMAT(/,' Enter filename to save cepstra data : ',$)
          CALL GETDFN(FILNAM)
          IF (OPNFIL(3,FILNAM,'W') .NE. .TRUE.) GOTO 402
          CALL PUTDAT(3,NFFT,TFCTR,CY)
C
C      Get original convolved X and Y data back into arrays
C
415     DO 100 I=1,NFFT
          CY(I)=AUX(I)
          Y(I)=(I-1)*TFCTR
100     CONTINUE
C
C      INVERSE PROCESSING
C
120     TYPE 121
121     FORMAT(/,' Do you want a symmetrical filter ',$)
        IANS = ASK('N')
        TYPE 133
133     FORMAT(/,' Enter data type (1 for impulse train, 2 for pulse ')
        TYPE 134
134     FORMAT(' 1 dates system impulse, 2 dates reference data) : ',$)
        ACCEPT *,LD
        NFF = (NFFT/2) - 1           ! set limit on time window
125     TYPE 126,NFF
126     FORMAT(/,' Enter positive time window ( < or = ',I3,' ) ')
        TYPE *,'Answer must be one half of total time window '
        TYPE 123
123     FORMAT(' desired for a symmetrical filter : ',$)
        ACCEPT *,IPOS
        IF(NFF .GE. IPOS) GOTO 517
        TYPE *,' '
        TYPE *,' Width selected exceeds positive time points in '
        TYPE *,' cepstrum !!! '
        TYPE *,' '
        GO TO 125
517     INEG=IPOS
        IF(IPOS .EQ .NFF) INEG = IPOS + 1
        IF(IANS .NE. .TRUE.) GOTO 520  ! Chose symmetrical filter
130       TYPE 131
131       FORMAT(/,' Enter  negative  time window (INTEGER): ',$)
          ACCEPT *,INEG
          INEG=IABS(INEG)
          IF(INEG .LE. NFFT/2) GOTO 520
          TYPE 132
132       FORMAT(' Width  exceeds negative time points in cepstrum ')
          GO TO 130
C
```

```
120     GW=INEG+IPOS       !TOTAL GATE USED
        GSTRT=-1.*FLOAT(INEG)
        GSTOP=FLOAT(IPOS)
140     GATENG=0.0
        IF(LO .EQ. 1) GOTO 150
C       Must have been pulse gate type selected
        ISTART=IPOS+1
        IEND=NFFT-INEG
        GO TO 170
C
C       LO=1 IMPLIES IMPULSE TRAIN RECOVERY
C
150     ISTART=1
        IEND=IPOS
        DO 160 I=ISTART,IEND
          GATENG=GATENG+CY(I)*CY(I)
          CY(I)=0.
160     CONTINUE
        ISTART=NFFT-INEG+1
        IEND=NFFT
170     DO 180 I=ISTART,IEND
          GATENG=GATENG+CY(I)*CY(I)
          CY(I)=0.
180     CONTINUE
        GATENG = TOTENG - GATENG
        TYPE 190,GW,GATENG
190     FORMAT(/,' Energy removed by gatewidth of ',I3,' points = ',
     *E12.5,' Joules')
C         Rotate data
            DO 205 I=1,NFFT/2
              J=I+NFFT/2
              TEMP=CY(I)
              CY(I)=CY(J)
              CY(J)=TEMP
              Y(J)=(I-1)*TFCTR
              Y(I)=(I-(NFFT/2+1))*TFCTR
205         CONTINUE
        TYPE 201
201     FORMAT(/,' Do you wish to plot gated cepstra ',$)
        IF (ASK('N') .EQ. .TRUE.) GOTO 421
            CALL CLEAR
            CALL PLOTY(CY,NFFT,3)          ! RT-11 PLOT
C           CALL PLOTXY(Y,CY,NFFT,3)       ! RT-11 PLOT
421     TYPE 420
420     FORMAT(/,' Do you wish to save gated cepstra data ',$)
        IF (ASK('N') .EQ. .TRUE.) GOTO 431
422       TYPE 425
425       FORMAT(/,' Enter filename to save gated cepstra data ',$)
          CALL GETDFN(FILNAM)
          IF (OPNFIL(3,FILNAM,'W') .NE. .TRUE.) GOTO 422
          CALL PUTDAT(3,NFFT,TFCTR,CY)
C       Rotate data
431         DO 220 I=1,NFFT/2
              J=I+NFFT/2
              TEMP=CY(I)
              CY(I)=CY(J)
              CY(J)=TEMP
220         CONTINUE
222     TYPE *,' '
        TYPE *,' ... Now computing IFT. Prepare to plot IFT. '
```

```
      TYPE *,' '
C
C     COMPUTE INVERSE CEPSTRUM
C
      CALL ICEPS(CY,ISNX,ISFX)
C
C   If 'sign' from cepstral processing is -1, multiply data by -1
C
      IF(ISNX .NE. -1) GOTO 660
      DO 670 I = 1,NFFT
        CY(I) = -CY(I)
670   CONTINUE
C
C   Perform inverse exponential weighting
C
660   IF(LO .EQ. 2) IST=1
      IF(LO .NE. 2) IST=IABS(ISFX)
      IF (EW .EQ. 1.0) GOTO 526
          FX=1.
          DO 235 I=IST,NFFT
            CY(I)=CY(I)*FX
            FX=FX/EW
235       CONTINUE
526   CONTINUE
C
      ISHFT=0
      IF(LO .EQ. 1) ISHFT=ISFX
      DO 240 I=1,NFFT
        Y(I)=(I-1+ISHFT)*TFCTR
240   CONTINUE
C
C    Plot inverse gated cepstrum
C
          CALL CLEAR
          CALL PLOTXY(Y,CY,NFFT,3)      ! RT-11 PLOT
          TYPE 702
702       FORMAT(/,' Do you want to plot data on the plotter ',$)
          IF (ASK('N') .EQ. .TRUE.) GOTO 703
          CALL PLOTXY(Y,CY,NFFT,1)
703   TYPE 450
450   FORMAT(/,' Do you wish to save deconvolved data ',$)
      IF (ASK('N') .EQ. .TRUE.) GOTO 461
452     TYPE 455
455     FORMAT(/,' Enter deconvolved data filename : ',$)
        CALL GETDFN(FILNAM)
        IF (OPNFIL(3,FILNAM,'W') .NE. .TRUE.) GOTO 452
      CALL PUTDAT(3,NFFT,TFCTR,CY)
461   DO 250 I=1,NFFT
        CY(I) = AUX(I)
250   CONTINUE
      TYPE 261
261   FORMAT(/,' Plot remaining component at this gate width ',$)
      IF (ASK('Y') .NE. .TRUE.) GOTO 290
      IF (LO .EQ. 2) GOTO 280
      LO = 2
      GOTO 140
280   LO = 1
      GOTO 140
C
290   TYPE 320
```

```
320     FORMAT(/,' Analyze cepstra data with a different date width ',$)
        IF (ASK('Y') .NE. .TRUE.) GOTO 300
        TYPE 330
330     FORMAT(/,'  ... Prepare to plot total cepstra data. ')
        GOTO 84
300     TYPE 310
310     FORMAT(/,' Wish to analyze another convolved data file ',$)
        IF (ASK('N') .NE. .TRUE.) GOTO 7
        CALL EXIT
        END
```

```
      SUBROUTINE FAST(B,N)
C
C     The subroutines used in this program were taken from:
C
C     'Programs for Digital Signal Processing'
C     IEEE Press, 1979
C     345 East 47 Strett, New York, NY 10017
C     Sponsored by the IEEE Acoustics, Speech, and
C                      Signal Processing Society
C     Lib. of Congress Cat. Card # 79-89028
C     IEEE Book # 0-87942-128-2 (paperback ver.)
C                # 0-87942-127-4 (hardback)
C     Also published by John Wiley & Sons, Inc.
C     Wiley Order # 0-471-05961-7 (paperback ver.)
C                   # 0-471-05962-5 (hardback)
C
C
      DIMENSION B(2)
      COMMON /CONS/PII,P7,P7TWO,C22,S22,PI2
      PII=4.*ATAN(1.)
      PI8=PII/8.
      P7=1./SQRT(2.)
      P7TWO=2.*P7
      C22=COS(PI8)
      S22=SIN(PI8)
      PI2=2.*PII
C
      DO 10 I=1,15
      M=I
      NT=2**I
      IF(N.EQ.NT)GO TO 20
10    CONTINUE
      WRITE (4,9999)
9999  FORMAT(' N -- NOT A POWER OF 2 ')
      STOP
20    N4POW=M/2
C
      IF(M-N4POW*2)40,40,30
30    NN=2
      INT=N/NN
      CALL FR2TR(INT,B(1),B(INT+1))
      GO TO 50
40    NN=1
C
```

```
50      IF(N4POW.EQ.0)GOTO 70
        DO 60 IT=1,N4POW
        NN=NN*4
        INT=N/NN
        CALL FR4TR(INT,NN,B(1),B(INT+1),B(2*INT+1),B(3*
     1INT+1),B(1),B(INT+1),B(2*INT+1),B(3*INT+1))
60      CONTINUE
C
70      CALL FORD1(M,B)
        CALL FORD2(M,B)
        T=B(2)
        B(2)=0.
        B(N+1)=T
        B(N+2)=0.
        DO 80 IT=4,N,2
        B(IT)=-B(IT)
80      CONTINUE
        RETURN
        END
C
C
C
        SUBROUTINE FORD1(M,B)
        DIMENSION B(2)
        K=4
        KL=2
        N=2**M
        DO 40 J=4,N,2
        IF(K-J)20,20,10
10      T=B(J)
        B(J)=B(K)
        B(K)=T
20      K=K-2
        IF(K-KL)30,30,40
30      K=2*J
        KL=J
40      CONTINUE
        RETURN
        END
C
C
C
        SUBROUTINE FORD2(M,B)
        DIMENSION L(15),B(2)
        EQUIVALENCE (L15,L(1)),(L14,L(2)),(L13,L(3)),(L12,L(4)),(L11,L(5))
     *,(L10,L(6)),(L9,L(7)),(L8,L(8)),(L7,L(9)),(L6,L(10)),(L5,L(11)),
     *(L4,L(12)),(L3,L(13)),(L2,L(14)),(L1,L(15))
C
        N=2**M
        L(1)=N
        DO 10 K=2,M
        L(K)=L(K-1)/2
10      CONTINUE
        DO 20 K=M,14
        L(K+1)=2
20      CONTINUE
        IJ=2
C
C
C
```

```
      DO 120 J1=2,L1,2
      DO 120 J2=J1,L2,L1
      DO 120 J3=J2,L3,L2
      DO 120 J4=J3,L4,L3
      DO 120 J5=J4,L5,L4
      DO 120 J6=J5,L6,L5
      DO 120 J7=J6,L7,L6
      DO 120 J8=J7,L8,L7
      DO 120 J9=J8,L9,L8
      DO 120 J10=J9,L10,L9
      DO 120 J11=J10,L11,L10
      DO 120 J12=J11,L12,L11
      DO 120 J13=J12,L13,L12
      DO 120 J14=J13,L14,L13
      DO 120 JI=J14,L15,L14
C
      IF(IJ-JI)30,120,120
30    T=B(IJ-1)
      B(IJ-1)=B(JI-1)
      B(JI-1)=T
      T=B(IJ)
      B(IJ)=B(JI)
      B(JI)=T
120   IJ=IJ+2
C
      RETURN
      END
C
C
C
C
      SUBROUTINE FR2TR(INT,B0,B1)
      DIMENSION B0(2),B1(2)
      DO 10 K=1,INT
      T=B0(K)+B1(K)
      B1(K)=B0(K)-B1(K)
      B0(K)=T
10    CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE FR4TR(INT,NN,B0,B1,B2,B3,B4,B5,B6,B7)
      DIMENSION L(15),B0(2),B1(2),B2(2),B3(2),B4(2),
     1B5(2),B6(2),B7(2)
      COMMON /CONS/PII,P7,P7TWO,C22,S22,PI2
      EQUIVALENCE (L15,L(1)),(L14,L(2)),(L13,L(3)),(L12,L(4)),(L11,L(5)
     *,(L10,L(6)),(L9,L(7)),(L8,L(8)),(L7,L(9)),(L6,L(10)),(L5,L(11)),
     *(L4,L(12)),(L3,L(13)),(L2,L(14)),(L1,L(15))
C
      L(1)=NN/4
      DO 40 K=2,15
      IF (L(K-1)-2)10,20,30
10    L(K-1)=2
20    L(K)=2
      GOTO 40
30    L(K)=L(K-1)/2
40    CONTINUE
C
```

```
      PIOVN=PI2/FLOAT(NN)
      JI=3
      JL=2
      JR=1
C
C
C
      DO 120 J1=2,L1,2
      DO 120 J2=J1,L2,L1
      DO 120 J3=J2,L3,L2
      DO 120 J4=J3,L4,L3
      DO 120 J5=J4,L5,L4
      DO 120 J6=J5,L6,L5
      DO 120 J7=J6,L7,L6
      DO 120 J8=J7,L8,L7
      DO 120 J9=J8,L9,L8
      DO 120 J10=J9,L10,L9
      DO 120 J11=J10,L11,L10
      DO 120 J12=J11,L12,L11
      DO 120 J13=J12,L13,L12
      DO 120 J14=J13,L14,L13
      DO 120 JTHET=J14,L15,L14
      TH2=JTHET-2
      IF(TH2)50,50,90
50    DO 60 K=1,INT
      T0=B0(K)+B2(K)
      T1=B1(K)+B3(K)
      B2(K)=B0(K)-B2(K)
      B3(K)=B1(K)-B3(K)
      B0(K)=T0+T1
      B1(K)=T0-T1
60    CONTINUE
C
      IF(NN-4)120,120,70
70    K0=INT*4+1
      KL=K0+INT-1
      DO 80 K=K0,KL
      PR=P7*(B1(K)-B3(K))
      PI=P7*(B1(K)+B3(K))
      B3(K)=B2(K)+PI
      B1(K)=PI-B2(K)
      B2(K)=B0(K)-PR
      B0(K)=B0(K)+PR
80    CONTINUE
      GOTO 120
C
90    ARG=TH2*PIOVN
      C1=COS(ARG)
      S1=SIN(ARG)
      C2=C1**2-S1**2
      S2=C1*S1+C1*S1
      C3=C1*C2-S1*S2
      S3=C2*S1+C1*S2
C
      INT4=INT*4
      J0=JR*INT4+1
      K0=JI*INT4+1
      JLAST=J0+INT-1
      DO 100 J=J0,JLAST
      K=K0+J-J0
```

```
      R1=B1(J)*C1-B5(K)*S1
      R5=B1(J)*S1+B5(K)*C1
      T2=B2(J)*C2-B6(K)*S2
      T6=B2(J)*S2+B6(K)*C2
      T3=B3(J)*C3-B7(K)*S3
      T7=B3(J)*S3+B7(K)*C3
      T0=B0(J)+T2
      T4=B4(K)+T6
      T2=B0(J)-T2
      T6=B4(K)-T6
      T1=R1+T3
      T5=R5+T7
      T3=R1-T3
      T7=R5-T7
      B0(J)=T0+T1
      B7(K)=T4+T5
      B6(K)=T0-T1
      B1(J)=T5-T4
      B2(J)=T2-T7
      B5(K)=T6+T3
      B4(K)=T2+T7
      B3(J)=T3-T6
100   CONTINUE
      JR=JR+2
      JI=JI-2
      IF(JI-JL)110,110,120
110   JI=2*JR-1
      JL=JR
120   CONTINUE
C
      RETURN
      END
C
C
C
      SUBROUTINE FSST(B,N)
      DIMENSION B(2)
      COMMON /CONST/PII,P7,P7TWO,C22,S22,PI2
      PII=4.*ATAN(1.)
      PI8=PII/8.
      P7=1./SQRT(2.)
      P7TWO=2.*P7
      C22=COS(PI8)
      S22=SIN(PI8)
      PI2=2.*PII
C
      DO 10 I=1,15
      M=I
      NT=2**I
      IF(N.EQ.NT)GO TO 20
10    CONTINUE
      WRITE (4,9999)
9999  FORMAT(' N -- NOT A POWER OF 2 ')
      STOP
20    B(2)=B(N+1)
      DO 30 I=4,N,2
      B(I)=-B(I)
30    CONTINUE
      DO 40 I=1,N
      B(I)=B(I)/FLOAT(N)
```

```
40      CONTINUE
C
        N4POW=M/2
C
        CALL FORD2(M,B)
        CALL FORD1(M,B)
        IF(N4POW.EQ.0)GOTO 60
        NN=4*N
        DO 50 IT=1,N4POW
        NN=NN/4
        INT=N/NN
        CALL FR4SYN(INT,NN,B(1),B(INT+1),B(2*INT+1)
     1,B(3*INT+1),B(1),B(INT+1),B(2*INT+1)
     1,B(3*INT+1))
50      CONTINUE
C
60      IF(M-N4POW*2)80,80,70
70      INT=N/2
        CALL FR2TR(INT,B(1),B(INT+1))
80      RETURN
        END
C
C
C
        SUBROUTINE FR4SYN(INT,NN,B0,B1,B2,B3,B4,B5,B6,B7)
        DIMENSION L(15),B0(2),B1(2),B2(2),B3(2),B4(2),B5(2),B6(2),B7(2)
        COMMON /CONST/PII,P7,P7TWO,C22,S22,PI2
        EQUIVALENCE (L15,L(1)),(L14,L(2)),(L13,L(3)),(L12,L(4)),(L11,L(5)
     *,(L10,L(6)),(L9,L(7)),(L8,L(8)),(L7,L(9)),(L6,L(10)),(L5,L(11)),
     *(L4,L(12)),(L3,L(13)),(L2,L(14)),(L1,L(15))
C
        L(1)=NN/4
        DO 40 K=2,15
        IF (L(K-1)-2)10,20,30
10      L(K-1)=2
20      L(K)=2
        GOTO 40
30      L(K)=L(K-1)/2
40      CONTINUE
C
        PIOVN=PII/FLOAT(NN)
        JI=3
        JL=2
        JR=2
C
C
C
        DO 120 J1=2,L1,2
        DO 120 J2=J1,L2,L1
        DO 120 J3=J2,L3,L2
        DO 120 J4=J3,L4,L3
        DO 120 J5=J4,L5,L4
        DO 120 J6=J5,L6,L5
        DO 120 J7=J6,L7,L6
        DO 120 J8=J7,L8,L7
        DO 120 J9=J8,L9,L8
        DO 120 J10=J9,L10,L9
        DO 120 J11=J10,L11,L10
        DO 120 J12=J11,L12,L11
        DO 120 J13=J12,L13,L12
```

```
      DO 120 J14=J13,L14,L13
      DO 120 JTHET=J14-L15,L14
      TH2=JTHET-2
      IF(TH2)50,50,90
50    DO 60 K=1,INT
      T0=B0(K)+B1(K)
      T1=B0(K)-B1(K)
      T2=B2(K)*2.0
      T3=B3(K)*2.0
      B0(K)=T0+T2
      B2(K)=T0-T2
      B1(K)=T1+T3
      B3(K)=T1-T3
60    CONTINUE
C
      IF(NN-4)120,120,70
70    K0=INT*4+1
      KL=K0+INT-1
      DO 80 K=K0,KL
      T2=B0(K)-B2(K)
      T3=B1(K)+B3(K)
      B0(K)=(B0(K)+B2(K))*2.0
      B2(K)=(B3(K)-B1(K))*2.0
      B1(K)=(T2+T3)*P7TWO
      B3(K)=(T3-T2)*P7TWO
80    CONTINUE
C
      GO TO 120
90    ARG=TH2*PIOVN
      C1=COS(ARG)
      S1=-SIN(ARG)
      C2=C1**2-S1**2
      S2=C1*S1+C1*S1
      C3=C1*C2-S1*S2
      S3=C2*S1+C1*S2
C
      INT4=INT*4
      J0=JR*INT4+1
      K0=JI*INT4+1
      JLAST=J0+INT-1
      DO 100 J=J0,JLAST
      K=K0+J-J0
      T0=B0(J)+B6(K)
      T1=B7(K)-B1(J)
      T2=B0(J)-B6(K)
      T3=B7(K)+B1(J)
      T4=B2(J)+B4(K)
      T5=B5(K)-B3(J)
      T6=B5(K)+B3(J)
      T7=B4(K)-B2(J)
      B0(J)=T0+T4
      B4(K)=T1+T5
      B1(J)=(T2+T6)*C1-(T3+T7)*S1
      B5(K)=(T2+T6)*S1+(T3+T7)*C1
      B2(J)=(T0-T4)*C2-(T1-T5)*S2
      B6(K)=(T0-T4)*S2+(T1-T5)*C2
      B3(J)=(T2-T6)*C3-(T3-T7)*S3
      B7(K)=(T2-T6)*S3+(T3-T7)*C3
100   CONTINUE
      JR=JR+2
```

```
      JI=JI-2
      IF(JI-JL)110,110,120
110   JI=2*JR-1
      JL=JR
120   CONTINUE
C
      RETURN
      END
```